NEW NUMERICAL AND COMPUTATIONAL METHODS LEVERAGING DYNAMICAL SYSTEMS THEORY FOR MULTI-BODY ASTRODYNAMICS

A Dissertation
Presented to
The Academic Faculty

By

Bhanu Kumar

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics

Georgia Institute of Technology

August 2022

NEW NUMERICAL AND COMPUTATIONAL METHODS LEVERAGING DYNAMICAL SYSTEMS THEORY FOR MULTI-BODY ASTRODYNAMICS

Thesis committee:

Dr. Rafael de la Llave, Advisor
School of Mathematics
Georgia Institute of Technology

Dr. Molei Tao
School of Mathematics
Georgia Institute of Technology

Dr. Rodney Anderson
Mission Design and Navigation Section
Jet Propulsion Laboratory, California Institute of Technology

Dr. Albert Fathi
School of Mathematics
Georgia Institute of Technology

Dr. Brian Gunter
School of Aerospace Engineering
Georgia Institute of Technology

Date approved: April 20, 2022

On sera frappé de la complexité de cette figure, que je ne cherche même pas à tracer. Rien n'est plus propre à nous donner un idée de la complication du problème des trois corps et en général de tous les problèmes de Dynamique où il n'y a pas d'intégrale uniforme

Henri Poincaré, "Les méthodes nouvelles de la mécanique céleste", 1892

To Dr. Kamla Saxena, my Nani Ma

ACKNOWLEDGMENTS

First of all, I would like to express my utmost gratitude and appreciation to my thesis advisor, Prof. Rafael de la Llave, for his support, guidance, encouragement, and patience during this entire process. Whether it be his seemingly infinite technical knowledge or his wealth of experience navigating life as a researcher, he has always been generous in sharing all of it, and I couldn't have asked for a better advisor! I consider myself very fortunate to have had the opportunity to academically "grow up" under his supervision. While discussing my good luck of having found such a good advisor, I must also thank Dr. Martin Lo, at the NASA Jet Propulsion Laboratory, for being the one who actually recommended to me as an undergraduate so strongly to go do my PhD with Rafael!

I would be remiss not to also give a very special thanks to my NASA research collaborator and mentor at the Jet Propulsion Laboratory, Dr. Rodney Anderson. Between the many hours spent with Rafael and I providing practical insight into problems and methods relevant to real-world applications, and discussing and helping implement future post-PhD plans, I am grateful to all his assistance in the last 4 years since we started working together. I look forward to joining him soon as a postdoc at JPL and continuing our collaboration.

I would like to acknowledge my thesis committee members, Profs. Brian Gunter, Albert Fathi, and Molei Tao, for their assistance during my graduate study. Thanks to Prof. Gunter for helping me extend my PhD minor into an aerospace MS degree, and serving as the aerospace research advisor for that. And a thank you to Prof. Fathi as well for funding my attendance at the summer school and conference in Venice in summer 2018, where I made several contacts who have developed over time into useful resources for my future.

In addition to my advisor, NASA mentor, and committee members, I must mention several other professors whose assistance was very valuable, both for research as well as professional advice. I would like to appreciate Profs. Alex Haro, Josep-Maria Mondelo, Angel Jorba, and Jay Mireles-James for their many useful discussions and assistance with my research. A special thanks also to Profs. Haro, Mireles-James, and Alessandra Celletti for their recommendations of me during my applications to different post-PhD positions. And a thank you also to the other members of our “Team Rafa” research group: Jiaqi Yang, Adrián Pérez Bustamante, Yian Yao, Jieun Seong, Jorge Gonzalez, and Chris Dupre.

Apart from all the people who have helped me professionally, the support of my friends has been crucial to making the last several years go smoothly and enjoyably. I am thankful to have had two longtime friends from my undergraduate days, Daniel Watts and Kevin Reilley, be able to accompany me on my PhD journey as well. I am happy to have met so many new friends among the other GT math grad students as well, including Hyun-Ki Min, Jaemin Park, Anubhav Mukherjee, Jaewoo Jung, Tom Rodewald, and José Acevedo, among others, to provide me company and levity during my days here. And there is no way I can ever forget the adventures I had with my dear MSRI gang during our time in Berkeley; thank you George “Georgy” Miloshevich, Victor “VVDR the Klein bottle” Vilaca da Rocha, Rosa “Rosita the chocolatier” Maria Vargas, and Nathan “the ’strayan” Duignan for making those 4 months some of the best ones of my life. I’m especially happy that I managed to trap you, VVDR, into coming to do a postdoc here at GT with Rafael.

Finally, these acknowledgments would be woefully incomplete without mentioning the unconditional support and love of my family both during and long before my graduate studies, without which I would never have reached where I am today.

The author gratefully acknowledges the support for this work provided by the National Aeronautics and Space Administration through NASA Space Technology Research Fellowship grant number 80NSSC18K1143, and partial travel support by European Commission grant H2020-MSCA-RISE-2015, Project 734557 (Trax). Part of the research presented in this dissertation has been carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Many proposed interplanetary space missions, including Europa Lander and Dragonfly, involve trajectory design in environments where multiple large bodies exert gravitational influence on the spacecraft, such as the Jovian and Saturnian systems as well as cislunar space. In these contexts, an analysis based on the mathematical theory of dynamical systems provides both better insight as well as new tools to use for the mission design compared to classic two-body Keplerian methods. Indeed, a rich variety of dynamical phenomena manifest themselves in such systems, including libration point dynamics, stable and unstable mean-motion resonances, and chaos. To understand the previously mentioned dynamical behaviors, invariant manifolds such as periodic orbits, quasi-periodic invariant tori, and stable/unstable manifolds are the major objects whose interactions govern the local and global dynamics of relevant celestial systems.

This work is focused on the development of numerical methodologies for computing such invariant manifolds and investigating their interactions. In Chapter 2, after a study of persistence of mean-motion resonances in the planar circular restricted 3-body problem (PCRTBP), techniques for computing the stable/unstable manifolds attached to resonant periodic orbits and heteroclinics corresponding to resonance transitions are presented. Chapter 3 focuses on the development of accurate and efficient parameterization methods for numerical calculation of whiskered quasi-periodic tori and their attached stable/unstable manifolds, for periodically-forced PCRTBP models. As part of this, a method for Levi-Civita regularization of such periodically-forced systems is introduced. Finally, Chapter 4 presents methods for combining the previously mentioned parameterizations with knowledge of the objects' internal dynamics, collision detection algorithms, and GPU computing to very rapidly compute propellant-free heteroclinic connecting trajectories between them, even in higher dimensional models. Such heteroclinics are key to the generation of chaos and large scale transport in astrodynamical systems.

CHAPTER 1
INTRODUCTION AND BACKGROUND

"One is struck by the complexity of this figure, which I will not even try to draw. Nothing is better to give us an idea of the complication of the problem of three bodies and, in general, of all the problems of dynamics where there is not a uniform integral." [1]

(Henri Poincaré, "Les méthodes nouvelles de la mécanique céleste", 1892)

As the 19th century was coming to a close, Henri Poincaré's research into the 3-body problem [1] led to the beginnings of the modern mathematical theory of dynamical systems. Yet, though the roots of the field lie in its applications to celestial mechanics 130 years ago, it is only in relatively recent years that increases in computing power, as well as new research and algorithms, have led to methods from dynamical systems theory seeing wider acceptance and interest for applications to space mission trajectory design. At a broad level, this work aims to to deepen and advance this encouraging trend, by investigating and adapting recent results and computational techniques from dynamical systems to unlock new capabilities for practical multi-body space mission design.

Since the beginning of the space age, the majority of mission trajectory design has been done using patched conic approximations, where the spacecraft motion is approximated as a series of two-body problems with the influencing central gravitational body changing depending on the spacecraft position; these classical methods are described in many text-books, for instance [2]. However, such methodologies fail to take advantage of the chaotic nature of multi-body celestial mechanics; indeed, the spacecraft dynamics can be extremely sensitive to initial conditions when influenced by multiple large bodies. This "butterfly effect" can in turn be used to move large distances with no or minimal fuel consumption, a very desirable outcome given the limited fuel capacity of deep-space probes.

The Genesis mission [3], launched in 2001, is an example of a real-life mission which used tools from dynamical systems theory for its mission design. In particular, Genesis was able to move between Sun-Earth L1 and Sun-Earth L2 without the use of any deterministic thrusting maneuvers. To accomplish this, the spacecraft leveraged heteroclinic connections between unstable periodic orbits near those libration points. These connections correspond to intersections of stable and unstable manifolds of periodic orbits, a phenomenon which generates a chaotic "tangle" leading to trajectories with wildly different itineraries starting arbitrarily close to each other. Indeed, this chaotic tangle is precisely the figure which Poincaré would not dare to draw, described in the quote at the start of this chapter.

Given the previous discussion, as well as other results on large-scale chaotic motions such as the Chirikov resonance overlap criterion [4], it is clear that stable and unstable manifolds, and their intersections, are key to understanding the dynamics of multi-body astrodynamical systems. These stable/unstable manifolds are themselves attached to either periodic orbits, as in the Genesis case, or to higher dimensional analogues of periodic orbits called quasi-periodic orbits, also known as invariant tori. The periodic/quasi-periodic orbits along with their stable/unstable manifolds and any heteroclinic connections form a kind of skeleton organizing the major dynamical behaviors present in the system. Hence, their study is of utmost importance, but to accomplish this, we need to be able to compute all of these objects in a computationally accurate and fast manner. This requirement is the focus of the work which will be presented in this dissertation.

1.1 Models and Equations of Motion

1.1.1 The Planar Circular Restricted 3-body Problem

The simplest dynamical model which still captures many of the interesting phenomena present in multi-body astrodynamics is the well-known planar circular restricted 3-body problem (PCRTBP). In the PCRTBP, one considers the motion of an infinitesimally small particle (thought of as a spacecraft) under the gravitational influence of two large bodies

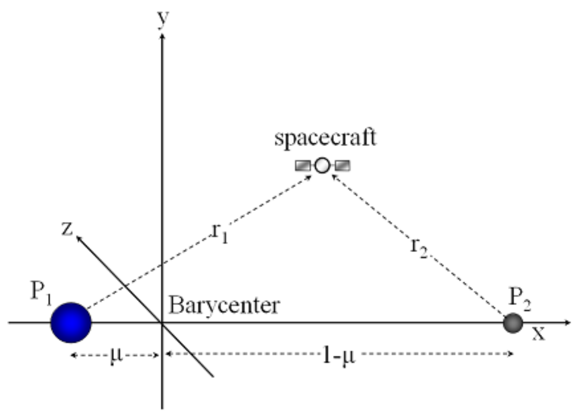


Figure 1.1: Diagram of Circular Restricted 3-Body Problem in Synodic Coordinate Frame [6]

of masses m_1 and m_2, collectively referred to as the primaries. It is assumed that m_1 and m_2 revolve about their common center of mass in a circular Keplerian orbit. Units are also normalized so that the distance between the two primaries becomes 1, $\mathcal{G}(m_1 + m_2)$ becomes 1, and their period of revolution becomes 2π. We define a mass ratio $\mu = \frac{m_2}{m_1+m_2}$, and unless otherwise specified, use a synodic, rotating non-inertial cartesian coordinate system centered at the barycenter of the primaries such that the two primaries are always on the x-axis. Due to the normalized units, the primary body will be at $x = -\mu$, and the secondary will be at $x = 1 - \mu$.

In the planar case we are studying here, we also assume that the spacecraft moves in the same orbit plane as the primaries. In this case, and in this synodic coordinate system, the equations of motion become [5]

$$\ddot{x} - 2\dot{y} = x - (1-\mu)\frac{x+\mu}{r_1^3} - \mu\frac{x-1+\mu}{r_2^3} \tag{1.1}$$

$$\ddot{y} + 2\dot{x} = y - (1-\mu)\frac{y}{r_1^3} - \mu\frac{y}{r_2^3} \tag{1.2}$$

where $r_1 = \sqrt{(x+\mu)^2 + y^2}$ is the distance from the spacecraft to m_1 and $r_2 = \sqrt{(x-1+\mu)^2 + y^2}$ is the distance to m_2. Fig. 1.1 is a diagram of the model, except for in our analysis we restrict ourselves to the case of $z = 0$.

There are two important properties of Eq. (1.1) and (1.2) to note. First of all, changing to position-momentum coordinates using $p_x = \dot{x} - y$ and $p_y = \dot{y} + x$, the equations of motion are Hamiltonian with form

$$\dot{x} = \frac{\partial H_0}{\partial p_x} \quad \dot{y} = \frac{\partial H_0}{\partial p_y} \qquad \dot{p}_x = -\frac{\partial H_0}{\partial x} \quad \dot{p}_y = -\frac{\partial H_0}{\partial y} \tag{1.3}$$

$$H_0(x, y, p_x, p_y) = \frac{p_x^2 + p_y^2}{2} + p_x y - p_y x - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{1.4}$$

Since the Hamiltonian in Eq. (1.4) is autonomous, it is an integral of motion. Hence, trajectories in the PCRTBP are restricted to 3-dimensional energy submanifolds of the state space satisfying $H_0(x, y, p_x, p_y) =$ constant. The Jacobi integral C, which is commonly encountered in the literature, is related to the Hamiltonian by $C = -2H_0$.

The second property to note is that the equations of motion have a time-reversal symmetry. Namely, if $(x(t), y(t), t)$ is a solution of Eq. (1.1) and (1.2) for $t > 0$, then $(x(-t), -y(-t), t)$ is a solution for $t < 0$.

1.1.2 Periodic Perturbations of the PCRTBP

The PCRTBP model exhibits many of the important dynamical phenomena present in multi-body celestial systems. However, there are many effects which are not included in the PCRTBP; many of these other influences on the spacecraft motion act in an approximately time-periodic manner, while preserving the Hamiltonian nature of the system. Here we will study dynamical models where one such periodic forcing effect is considered in addition to the PCRTBP. The equations of motion in this case are given by Eq. (1.5) along with time-periodic Hamiltonian (1.6)

$$\dot{x} = \frac{\partial H_\varepsilon}{\partial p_x} \quad \dot{y} = \frac{\partial H_\varepsilon}{\partial p_y} \qquad \dot{p}_x = -\frac{\partial H_\varepsilon}{\partial x} \quad \dot{p}_y = -\frac{\partial H_\varepsilon}{\partial y} \qquad \dot{\theta}_p = \Omega_p \tag{1.5}$$

$$H_\varepsilon(x, y, p_x, p_y, \theta_p) = H_0(x, y, p_x, p_y) + H_1(x, y, p_x, p_y, \theta_p; \varepsilon) \tag{1.6}$$

where $\theta_p \in \mathbb{T}$ is an angle, H_0 is the PCRTBP Hamiltonian given by Eq. (1.4), H_1 is the perturbation by the time-periodic effect and satisfies $H_1(x, y, p_x, p_y, \theta_p; 0) = 0$, and $\varepsilon > 0$ and Ω_p are the perturbation parameter and perturbation frequency, respectively. ε signifies the strength of the perturbation, $\varepsilon = 0$ being the unperturbed PCRTBP, and Ω_p is a known constant frequency. The perturbation from H_1 is $2\pi/\Omega_p$ periodic, with θ_p being the perturbation phase angle. In general, the Hamiltonian in Eq. (1.6) is not an integral of motion when $\varepsilon \neq 0$.

There are many different Hamiltonian periodically perturbed PCRTBP models of interest for applications. A common perturbation added to the PCRTBP is that of a third large body revolving in a circle (or approximate circle) around m_1 or m_2. Examples of these restricted 4-body models include the bicircular problem [7], the coherent quasi-bicircular problem [8], and the Hill restricted 4-body problem [9]. Another common periodically-perturbed PCRTBP model is the planar elliptic RTBP (PERTBP).

1.1.3 The Planar Elliptic Restricted 3-body Problem

The tools we develop in Chapters 3 and 4 of this dissertation are applicable to a wide variety of Hamiltonian periodic perturbations as discussed in the previous section. However, in this dissertation, we use the PERTBP for numerical demonstration of their usage. In the PERTBP, m_1 and m_2 revolve around their barycenter in an elliptical Keplerian orbit of nonzero eccentricity $\varepsilon > 0$. All other assumptions are the same as the PCRTBP. The length unit is normalized such that the m_1-m_2 orbit semi-major axis is 1, and the period of the primaries' orbit remains 2π. This implies that the perturbation frequency $\Omega_p = 1$; hence, we can take $\theta_p = t$ modulo 2π.

The PERTBP model we use is essentially the same as that used by [10], except for a transformation from position-velocity to position-momentum coordinates and a restriction to the xy-plane. The coordinate system is again such that m_1 and m_2 are always on the x-axis with the origin at their barycenter. However, the distance between them is now time-periodic, with periapse at $t = 0$; this is different from the pulsating coordinates used by [11]. The equations of motion are Eq. (1.5) with time-periodic Hamiltonian

$$H_\varepsilon(x, y, p_x, p_y, t) = \frac{p_x^2 + p_y^2}{2} + n(t)(p_x y - p_y x) - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{1.7}$$

where we have $n(t) = \frac{\sqrt{1-\varepsilon^2}}{(1-\varepsilon\cos E(t))^2}$, $r_1 = \sqrt{(x + \mu(1-\varepsilon\cos E(t)))^2 + y^2}$ and $r_2 = \sqrt{(x - (1-\mu)(1-\varepsilon\cos E(t)))^2 + y^2}$. $E(t)$ is the 2π-periodic eccentric anomaly of the elliptical m_1-m_2 orbit, and can be computed by solving the well-known Kepler's equation $M = E - \varepsilon \sin E$ using methods such as those in [2]. $n(t)$ is the time derivative of the m_1-m_2 true anomaly. From Eq. (1.5) and (1.7), we have $p_x = \dot{x} - n(t)y$ and $p_y = \dot{y} + n(t)x$.

1.2 The Parameterization Method for Invariant Manifolds

The parameterization method is a general technique for the computation of many kinds of invariant objects in dynamical systems, including invariant tori as well as stable and unstable manifolds of fixed points, periodic orbits, and tori. It works in both Hamiltonian as well as in non-Hamiltonian systems, although as we will see in Chapter 3, it is often possible to take advantage of the special properties of the Hamiltonian case to enhance the implementation. Haro et al. [12] describe several applications of this method. The essential idea is that given a map $F : M \to M$ where M is some manifold, if we know that there is an F-invariant object diffeomorphic to some model manifold $\mathcal{M}$, then we can solve for an injective immersion $W : \mathcal{M} \to M$ and a diffeomorphism $f : \mathcal{M} \to \mathcal{M}$ such that the invariance equation

$$F(W(s)) = W(f(s)) \tag{1.8}$$

holds for all $s \in \mathcal{M}$. W is referred to as the parameterization of the invariant manifold, and f as the internal dynamics on the model manifold $\mathcal{M}$. Eq. (1.8) means that F maps the image $W(\mathcal{M})$ into itself, so that $W(\mathcal{M})$ is the invariant object in the full ambient space M. In this dissertation, we will always have $M = \mathbb{R}^4$.

1.3 Organization of this Dissertation

This thesis is based on a series of joint works with Dr. Rodney Anderson and Prof. Rafael de la Llave. In Chapter 2, which is based on the paper [13], after investigating the persistence of resonant periodic orbits in the PCRTBP, we leverage parameterization methods for accurate computation of their stable/unstable manifolds and heteroclinic connections. Chapter 3, based on [14], then considers the higher dimensional and more complex case of periodically-forced PCRTBP models, developing methods for efficiently computing whiskered tori (which unstable PCRTBP periodic orbits persist as) along with their center, stable, and unstable bundles and stable/unstable manifolds. Finally, Chapter 4 presents new algorithms for using manifold parameterizations and modern GPU-computing hardware for very fast searches and accurate computation of heteroclinic connections between tori, in spite of the high dimensionality involved. Most of Chapter 4 is based on [15].

CHAPTER 2
HIGH-ORDER RESONANT ORBIT MANIFOLD EXPANSIONS FOR MISSION DESIGN IN THE PCRTBP

2.1 Introduction

In recent years, resonant periodic orbits and their stable and unstable manifolds have seen significant interest and use as a tool for trajectory design in multi-body systems. For instance, Anderson and Lo [16] demonstrated that a planar version of a Europa Orbiter trajectory designed in 1999 at JPL closely followed stable and unstable manifolds of unstable resonant periodic orbits during resonance transition. They also demonstrated [17] the development of new trajectories using homoclinic and heteroclinic connections between resonances. Resonant orbit manifold arcs were also used by Vaquero and Howell [18] to design transfers from LEO to Earth-Moon libration point orbits. More recently, out of the nine Titan-to-Titan encounters made by Cassini between July 2013 and June 2014, eight of the nine resulting transfers involved resonances [19]. And even more recently, the baseline mission design for the Europa Lander mission concept made profitable use of these mechanisms for the final approach to the surface of Europa [20]. For many other examples of applications of resonant orbits, see Anderson, Campagnola, and Lantoine [21].

However, the methods used in the previously mentioned studies, as well as in others, rely on using eigenvectors of the linearized dynamics as local approximations of the manifolds. Since such approximations are not accurate except very close to the base invariant object, this requires large amounts of numerical integration to globalize the manifolds and locate intersections, which can decrease accuracy as integration errors add up over longer integration times.

In this chapter, we study hyperbolic resonant periodic orbits in the planar circular restricted 3-body problem, and develop methods for accurately computing their manifolds and transfer trajectories between them. We first use the standard Melnikov method [22] to find Keplerian periodic orbits which survive for small values of the mass parameter μ. This perturbative analysis is followed by numerical continuation to compute the orbits for physically relevant μ values. We then implement the parameterization method [23, 12] to compute high order polynomial approximations of the stable and unstable manifolds. Finally, we develop an efficient method which combines the previously computed polynomials with a Poincaré section and bisection to compute heteroclinic connections. We also demonstrate application of these tools to the problem of transferring between resonances in the Jupiter-Europa system.

2.1.1 Delaunay and Synodic Delaunay Coordinates

The PCRTBP model described in Section 1.1 admits a change of coordinates from $(x, y, \dot{x}, \dot{y})$ to action angle coordinates,, which will be required for the first-order Melnikov analysis carried out in Section 2.2 (all other computations in this study will be done in the synodic cartesian coordinate frame). We summarize Celletti [5] here. Consider an inertial reference frame centered at the primary body m_1, and let $m_2 = 0$. Recall that the planar two-body problem in this coordinate frame can be expressed in Delaunay coordinates (L_0, G_0, ℓ_0, g_0), which are closely related to the classical orbital elements. For a two-body orbit, angle ℓ_0 is the mean anomaly, angle g_0 is the longitude of periapsis, and actions L_0 and G_0 are related to the semi-major axis a and eccentricity e as follows:

$$L_0 = \sqrt{a} \qquad G_0 = L_0\sqrt{1 - e^2} \tag{2.1}$$

Other texts generally write $L_0 = \sqrt{\mathcal{G}m_1 a}$; however with our normalized units, $\mathcal{G}m_1 = 1$ in the 2-body problem. In these coordinates, it can be shown that the evolution of

(L_0, G_0, ℓ_0, g_0) is Hamiltonian with Hamiltonian function

$$H(L_0, G_0, \ell_0, g_0) = -\frac{1}{2L_0^2} \tag{2.2}$$

and satisfies Hamilton's equations of motion

$$\frac{dL_0}{dt} = -\frac{\partial H}{\partial \ell_0} = 0 \qquad \frac{dG_0}{dt} = -\frac{\partial H}{\partial g_0} = 0 \tag{2.3}$$

$$\frac{d\ell_0}{dt} = \frac{\partial H}{\partial L_0} \qquad \frac{dg_0}{dt} = \frac{\partial H}{\partial G_0} \tag{2.4}$$

As expected, the actions are constant along trajectories, while only the angles (in fact, only ℓ_0) vary. If one now introduces a second large body of $\mathcal{G}m_2 = \mu$, then the system Hamiltonian Eq. (2.2) becomes

$$H(L_0, G_0, \ell_0, g_0) = -\frac{1}{2L_0^2} + \mu H_1(L_0, G_0, \ell_0, g_0, t) \tag{2.5}$$

where the perturbation $H_1(L_0, G_0, \ell_0, g_0, t)$ is

$$H_1(L_0, G_0, \ell_0, g_0, t) = \frac{r_1 \cos(\theta - t)}{\rho_2^2} - \frac{1}{\sqrt{\rho_2^2 + r_1^2 - 2\rho_2 r_1 \cos(\theta - t)}} \tag{2.6}$$

The quantity ρ_2 is the constant distance from m_2 to m_1; with our normalized units, $\rho_2 = 1$. r_1 as defined earlier is the distance from the spacecraft to m_1. $\theta = g_0 + f$ is the longitude of the spacecraft, where f is the spacecraft instantaneous true anomaly. Note that r_1 and f are functions of L_0, G_0, and ℓ_0.

Now, make a time-varying canonical change of variables $(L, G, \ell, g) = (L_0, G_0, \ell_0, g_0 - t)$; the new variable g is the instantaneous longitude of periapsis of the spacecraft orbit relative to the x-axis of the the synodic cartesian coordinate frame. Then, the Hamiltonian function from Eq. 2.5 and 2.6 becomes

$$H(L,G,\ell,g) = -\frac{1}{2L^2} - G + \mu H_1(L,G,\ell,g) \tag{2.7}$$

$$H_1(L,G,\ell,g) = \frac{r_1\cos(g+f)}{\rho_2^2} - \frac{1}{\sqrt{\rho_2^2 + r_1^2 - 2\rho_2 r_1 \cos(g+f)}} \tag{2.8}$$

which is no longer time-varying. We henceforth refer to these new coordinates as synodic Delaunay coordinates. Note that in these coordinates, for $\mu = 0$, the actions L and G are constant on trajectories, but

$$\frac{d\ell}{dt} = \frac{\partial H}{\partial L} = \frac{1}{L^3} = a^{-3/2} \qquad \frac{dg}{dt} = \frac{\partial H}{\partial G} = -1 \tag{2.9}$$

Since both angles are varying with time, even for $\mu = 0$ (m_2 infinitesimal, the two-body problem), not all orbits are periodic in these synodic Delaunay coordinates. Only orbits such that $k_1 a^{-3/2} + k_2(-1) = 0$ for some $k_1, k_2 \in \mathbb{Z}$ will be periodic, with period $2\pi k_1$. Note that $a^{-3/2}$ is the mean motion of the spacecraft, and 1 is the mean motion of m_2. Hence, for $\mu = 0$, an orbit in these coordinates is periodic if and only if the mean motions of the spacecraft and m_2 are rational multiples of each other. This is equivalent to there being $n, m \in \mathbb{Z}$ such that in the inertial reference frame, the spacecraft makes n revolutions around m_1 in the time that m_2 makes m revolutions around m_1. In the two-body problem ($\mu = 0$), such orbits are defined as $n : m$ resonant periodic orbits.

2.2 Persistence of Resonant Periodic Orbits

As described in Section 2.1.1, for $\mu = 0$, in synodic Delaunay coordinates, the only periodic orbits are $n : m$ resonant periodic orbits, $n, m \in \mathbb{Z}$. We are now interested in seeing which of these periodic orbits survive the perturbation when $\mu > 0$. For this, the perturbative method of Melnikov [22] is useful here. Without going into a full derivation, the essential theory is that given a periodic orbit $\mathbf{x}_0(t)$ in the $\mu = 0$ system, we can express solutions of the μ-dependent equations of motion (with initial condition $\mathbf{x}_\mu(0) = \mathbf{x}_0(0)$) as an expansion in powers of μ

$$\mathbf{x}_\mu(t) = \mathbf{x}_0(t) + \mu \mathbf{x}_1(t) + O(\mu^2) \tag{2.10}$$

where $\mathbf{x}_\mu(t) = (L(t,\mu), G(t,\mu), \ell(t,\mu), g(t,\mu))$.

Denote the period of $\mathbf{x}_0(t)$ by $T = 2\pi m$. The main conclusion of the Melnikov theory is that if an initial condition $\mathbf{x}_0(0) = (L_i, G_i, \ell_i, g_i)$ can be found such that $\mathbf{x}_1(T) = \mathbf{x}_1(0)$ in the perturbative expansion Eq. (2.10), then a true periodic orbit can be found near $\mathbf{x}_0(0)$ for μ small enough. This means that we can expect to be able to continue the $\mu = 0$ periodic orbit $\mathbf{x}_0(t)$ into $\mu > 0$. Furthermore, if one fixes L_i and G_i, and also (without loss of generality) sets $\ell_i = 0$, it can be shown that $\mathbf{x}_1(T) = \mathbf{x}_1(0)$ if and only if the Melnikov function

$$M(g_i) = \int_0^{2\pi m} \left(\frac{\partial H_0}{\partial \ell}\frac{\partial H_1}{\partial L} - \frac{\partial H_0}{\partial L}\frac{\partial H_1}{\partial \ell} \right)(L_i, G_i, \Omega(L_i)t, g_i - t)\, dt \tag{2.11}$$

$$= \int_0^{2\pi m} -\frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i, G_i, \frac{1}{L_i^3}t, g_i - t)\, dt \tag{2.12}$$

has simple zeros. If one of those zeros is at $g_i = g_{i,z}$, then we know that the periodic orbit with initial condition $\mathbf{x}_0(0) = (L_i, G_i, 0, g_{i,z})$ persists in a perturbed form for $\mu > 0$, albeit with a possibly slightly different period. Hence, one studies the Melnikov function $M(g_i)$ given in Eq. (2.11). Note that in the integral for $M(g_i)$, the integration of $\frac{\partial H_1}{\partial \ell}$ occurs only along the original, unperturbed periodic orbit.

One property of $M(g_i)$ is that it is an odd function, $M(g_i) = -M(-g_i)$. To show this, first note that

$$H_1(L, G, \ell, g) = H_1(L, G, -\ell, -g) \tag{2.13}$$

which then implies that

$$\begin{aligned}\frac{\partial H_1}{\partial \ell}(L,G,\ell,g) &= \frac{\partial}{\partial \ell}\left[H_1(L,G,-\ell,-g)\right] \\ &= -\frac{\partial H_1}{\partial \ell}(L,G,-\ell,-g)\end{aligned} \tag{2.14}$$

Hence, we find that (using $s=-t$ below)

$$\begin{aligned} M(g_i) &= \int_0^{2\pi m} -\frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i,G_i,\frac{1}{L_i^3}t, g_i-t)\,dt \\ &= \int_0^{-2\pi m} \frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i,G_i,-\frac{1}{L_i^3}s, g_i+s)\,ds \\ &= \int_{-2\pi m}^{0} -\frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i,G_i,-\frac{1}{L_i^3}s, g_i+s)\,ds \\ (*) &= \int_0^{2\pi m} -\frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i,G_i,-\frac{1}{L_i^3}s, g_i+s)\,ds \\ (**) &= \int_0^{2\pi m} \frac{1}{L_0^3}\frac{\partial H_1}{\partial \ell}(L_i,G_i,\frac{1}{L_i^3}s, -g_i-s)\,ds \\ &= -M(-g_i)\end{aligned} \tag{2.15}$$

where line $(*)$ is because $(L_i,G_i,-\frac{1}{L_i^3}s,g_i+s)$ is a $2\pi m$-periodic orbit, and the line $(**)$ follows from Eq. (2.14). Hence, we have proven that $M(g_i)$ is odd, and therefore has a zero at $g_i=0$.

We plotted $M(g_i)$ for several different resonances $n:m$. An example of such a plot is shown in Fig. 2.1 for $n=3$, $m=4$, (a 3:4 resonant periodic orbit) with eccentricity $e=0.5$.

One thing to note is that $M(g_i)$ is $2\pi/n$ periodic when we take n,m coprime. This periodicity is always present, as for an $n:m$ resonant orbit the mean anomaly $\ell=\frac{1}{L_i^3}t$ is $2\pi m/n$ periodic. So, evolving the point $(L_i,G_i,\ell=0,g_i)$ from $t=0$ to $t=2\pi m/n$ gives the point $(L_i,G_i,\ell=0,g_i-2\pi\frac{m}{n})$. Both points lie on the same periodic orbit, and so integrating $\frac{\partial H_1}{\partial \ell}$ from $t=0$ to $2\pi m$ along the orbit starting from either point gives the same final result. Integrating starting from the former point corresponds to $M(g_i)$, while starting

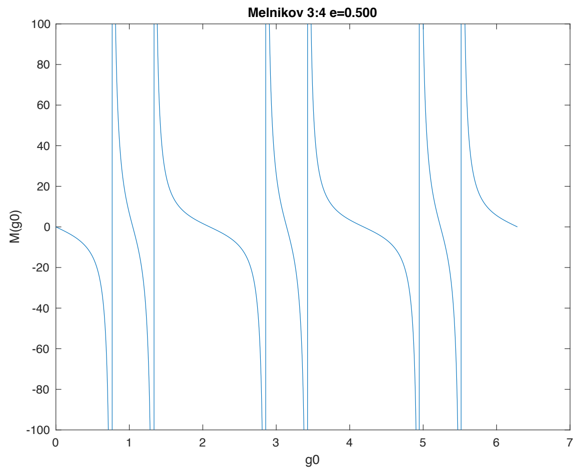


Figure 2.1: Plot of $M(g_i)$ for 3:4 resonance, $e = 0.5$

from the latter corresponds to $M(g_i - 2\pi\frac{m}{n})$; hence $M(g_i) = M(g_i - 2\pi\frac{m}{n})$. Since n, m are coprime, this implies $M(g_i) = M(g_i + \frac{2\pi}{n})$.

However, as is clear from the previous explanation, this periodicity gives us no additional useful zeros of $M(g_i)$; zeros differing by the quantity $2\pi/n$ are merely different points on the same orbit, and therefore do not correspond to different persistent resonant orbits. Hence, one can restrict the search for $M(g_i) = 0$ to the interval $g_i \in [0, 2\pi/n)$. Across many different values of n and m, apart from $g_i = 0$, the only other zero found for all tested cases was $g_i = \frac{\pi}{n}$. We have not analytically proven that $M(\pi/n) = 0$ for arbitrary m, n, but the numerical evidence is strong.

In summary, we have found that the two relevant zeros of $M(g_i)$ for an $n : m$ resonant periodic orbit are $g_i = 0$ and $g_i = \frac{\pi}{n}$. Hence, for $\mu > 0$ small enough, it should be possible to find periodic orbits close to the Keplerian orbits with initial conditions $(L_i, G_i, \ell = 0, g = 0)$ and $(L_i, G_i, \ell = 0, g = \pi/n)$, where $L_i = \sqrt{a}$ should be chosen so that the corresponding Keplerian orbit period satisfies the $n : m$ resonance condition; G_i should satisfy $0 < G_i < L_i$. Furthermore, as a consequence of the Poincaré-Birkhoff fixed point theorem [24], one of these two orbits will have elliptic stability type and the other should have hyperbolic stability. Intuitively, one expects the orbit corresponding to initial conditions $(L_i, G_i, \ell = 0, g = 0)$ to be the unstable, hyperbolic orbit, as this corresponds to the initial argument of periapse being aligned with a close flyby of m_2. It is on resonant orbits of this type that we concentrate now.

2.3 Computation of Resonant Periodic Orbits

With the persisting Keplerian resonant periodic orbits found, we next compute these surviving orbits and their periods for the full PCRTBP with physically relevant values of $\mu > 0$. Namely, for the Jupiter-Europa system we use $\mu_E = 2.5266448850435028 \times 10^{-5}$, and for Earth-Moon we used $\mu_M = 1.2150584270571545 \times 10^{-2}$. To this end, a continuation method was used, whereby the periodic orbits computed for smaller values of μ are used to find an initial guess for the periodic orbit and period corresponding to a larger value of μ.

We start with a value of μ for which we wish to compute an $n : m$ resonant orbit. We set $\mu_0 = 0$, $\mu_1 = \mu/N$, $\ldots$, $\mu_k = k\mu/N$, $\ldots$, $\mu_N = \mu$. We then seek to compute periodic points $\mathbf{x}_{\mu_k}$ and periods T_{sc,μ_k} corresponding to the PCRTBP periodic orbit for mass ratio value μ_k. $\mathbf{x}_{\mu_0}$ and $T_{sc,\mu_0} = 2\pi m$ are known from the Melnikov analysis; to simplify the computations, we convert the initial condition $\mathbf{x}_{\mu_0} = (L_i, G_i, \ell = 0, g = 0)$ back to the synodic cartesian coordinate frame $(x_i, y_i, \dot{x}_i, \dot{y}_i)$ and carry out subsequent computations in that frame.

To compute the $\mathbf{x}_{\mu_k}$ and T_{sc,μ_k}, we

1. Form an initial guess for $(\mathbf{x}_{\mu_k}, T_{sc,\mu_k})$ as

$$(\mathbf{x}_{\mu_k}, T_{sc,\mu_k})_{guess} = (\mathbf{x}_{\mu_{k-1}}, T_{sc,\mu_{k-1}}) + [(\mathbf{x}_{\mu_{k-1}}, T_{sc,\mu_{k-1}}) - (\mathbf{x}_{\mu_{k-2}}, T_{sc,\mu_{k-2}})] \quad (2.16)$$

   except if $k = 1$, $(\mathbf{x}_{\mu_1}, T_{sc,\mu_1})_{guess} = (\mathbf{x}_{\mu_0}, T_{sc,\mu_0})$.

2. Solve for $(\mathbf{x}_{\mu_k}$, $T_{sc,\mu_k})$ using initial guess and the MATLAB function fsolve on the equation

$$\Phi_{T_{sc,\mu_k}}(\mathbf{x}_{\mu_k}) - \mathbf{x}_{\mu_k} = 0 \quad (2.17)$$

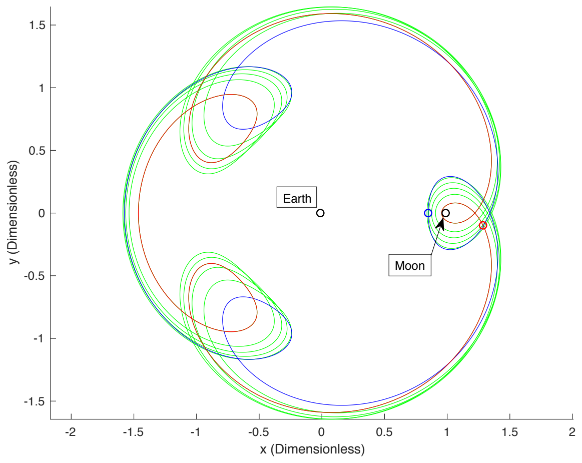


Figure 2.2: Continuation of 3:4, $e = 0.3, g_0 = 0$ resonant orbit from $\mu = 0$ (blue) to $\mu = \mu_M$ (red) with orbits for intermediate μ values shown in green.

where $\Phi_{T_{sc,\mu_k}}(\mathbf{x}_{\mu_k})$ denotes the flow of $\mathbf{x}_{\mu_k}$ by the equations of motion 1.1 and 1.2 by time T_{sc,μ_k}.

3. Increase k by 1, and return to step 1 until $k = N$.

Note that T_{sc,μ_k} must be allowed to vary in order to find periodic orbits for $\mu > 0$. Also, the solution of Eq. (2.17) is not unique for a given μ_k, as the value of the Hamiltonian (Eq. (1.4)) is not fixed, nor is there a condition added to fix the phasing of the point on a given periodic orbit. Nevertheless, the continuation was successful in continuing 100 different Keplerian resonant periodic orbits to $\mu = \mu_E$, and 32 different orbits to $\mu = \mu_M$. We conjecture that for a given resonance $n : m$ (and hence fixed semi-major axis a), continuation of orbits with different values of eccentricity e yields final orbits at different values of the Jacobi constant which can be computed from each other through continuation by energy. Additionally, do note that there exist resonant periodic orbits for $\mu > 0$ which are not continuations of $\mu = 0$ orbits [21].

An example of the continuation of a 3:4 resonant orbit with $e = 0.3$ in the Earth-Moon system is shown in Fig. 2.2. The blue curve is the original Keplerian periodic orbit. The red curve is the final computed periodic orbit for $\mu = \mu_M$, and the green curves are computed orbits for some intermediate μ values.

2.4 Parameterization of Invariant Manifolds

With the resonant periodic orbits and their periods computed for physically relevant values of μ, we next turn our attention to accurate computation of the orbits' stable and unstable invariant manifolds. As mentioned in the introduction, generally current studies using manifolds use linear approximations of invariant manifolds found by computing eigenvectors of the monodromy matrix of the periodic orbit. However, in our case, we compute high order (degree 25 to 50) Taylor polynomials which approximate the manifolds very accurately within some domain of validity.

Consider a hyperbolic resonant periodic orbit in the PCRTBP containing periodic point $\mathbf{x}_\mu$ and of period $T_{sc,\mu}$. To simplify computations, instead of considering the equations of motion 1.1 and 1.2, we instead consider the map $F : \mathbb{R}^4 \to \mathbb{R}^4$ defined as the time-$T_{sc,\mu}$ mapping by the equations of motion; using the notation established in the previous section, this simply means $F(\mathbf{x}) = \Phi_{T_{sc,\mu}}(\mathbf{x})$.

We know that $\mathbf{x}_\mu$ is a fixed point of the map F, and hence the monodromy matrix $DF(\mathbf{x}_\mu)$ represents the linearized dynamics around $\mathbf{x}_\mu$. Since we are looking at a hyperbolic periodic orbit, $DF(\mathbf{x}_\mu)$ has one stable and one unstable eigenvalue, in addition to two expected unit eigenvalues. Hence, we know that the stable and unstable manifolds of the fixed point $\mathbf{x}_\mu$ of the full nonlinear map F will also be 1-dimensional. Note that if we consider the full continuous-time flow and the periodic orbit, rather than the map F and its fixed point $\mathbf{x}_\mu$, the stable and unstable manifolds of the periodic orbit are 2-D. Specifically, they are cylinders corresponding to the well-known "tube dynamics" [25]. The stable and unstable manifolds of $\mathbf{x}_\mu$ under F will just be non-closed curves contained on the surface of these cylinders; by integrating points from these curves by the equations of motion, one can compute all the points on the cylindrical manifolds of the periodic orbit.

Remember that motion in our system is restricted to 3-D submanifolds of the state space corresponding to energy level sets; hence, a given periodic orbit and its stable and unstable manifolds will all be contained within a 3-D submanifold. If we have two periodic orbits at the same energy level, then the 2-D unstable manifold of the first orbit and the 2-D stable manifold of the second orbit will also be contained within a 3-D submanifold. Hence, if the manifolds intersect, they will generically intersect along a curve corresponding to a heteroclinic trajectory. Our final goal is to compute these heteroclinic connections between orbits.

However, computing 2-D manifolds of periodic orbits and their intersections requires significantly more computational tools and power than for 1-D manifolds of fixed points. Hence, we reduce the dimensionality of our problem through two steps. First of all, we compute 1-D stable and unstable manifolds of the fixed point $\mathbf{x}_\mu$ of the map F, rather than 2-D manifolds of orbits. Second, we take a Poincaré surface of section (a 2-D submanifold of the 3-D energy submanifold) passing through $\mathbf{x}_\mu$ and compute the 1-D intersection of the 2-D stable and unstable manifolds with the surface of section; this is simply done by propagating points from the 1-D manifolds of the fixed point $\mathbf{x}_\mu$ until their closest intersection with the section. These 1-D intersections of the periodic orbit manifolds with the surface of section simply correspond to stable and unstable manifolds of the fixed point $\mathbf{x}_\mu$ under the Poincaré return map.

2.4.1 The Parameterization Method for Invariant Manifolds

The parameterization method, as described in Section 1.2, is a general technique in dynamical systems useful for the computation of several types of invariant geometric structures. In this case, we seek to parametrize the 1-dimensional stable and unstable manifolds of the fixed point $\mathbf{x}_\mu$ of F. Hence, continuing in the framework of Section 1.2, the ambient manifold $M = \mathbb{R}^4$, the model manifold $\mathcal{M} = \mathbb{R}$, and furthermore we can take $f(s) = \lambda s$, where λ is the stable or unstable eigenvalue of $DF(\mathbf{x}_\mu)$, depending on which manifold we are trying to compute. Thus, the equation to solve for the parameterization $W(s)$ is

$$F(W(s)) - W(\lambda s) = 0 \tag{2.18}$$

where $s \in \mathbb{R}$. We express W as a Taylor series

$$W(s) = \mathbf{x}_\mu + \sum_{k \geq 1} W_k(s) \tag{2.19}$$

where $W_k(s)$ is a monomial of degree k in s. The constant term in W is $\mathbf{x}_\mu$, and the linear terms will be the stable or unstable eigenvector of $DF(\mathbf{x}_\mu)$ (we take unit length eigenvectors). Hence we need to solve for the higher-order terms $W_k(s)$, $k \geq 2$.

Denote $W_{<k}(s) = \mathbf{x}_\mu + \sum_{j=1}^{k-1} W_j(s)$. Assume that we have solved for all $W_j(s)$ for $j < k$, so that $F(W_{<k}(s)) - W_{<k}(\lambda s)$ has only s^k and higher order terms. Then, the method to solve for $W_k(s)$ is:

1. Find $E_k(s) = [F(W_{<k}(s)) - W_{<k}(\lambda s)]_k$, where $[\cdot]_k$ denotes the s^k term of the RHS.

2. Solve for the s^k term $W_k(s)$ which when added to $W_{<k}(s)$ cancels $E_k(s)$ in Eq. (2.18).

$$\begin{aligned} -E_k(s) &= DF(\mathbf{x}_\mu) W_k(s) - W_k(\lambda s) \\ &= \left[DF(\mathbf{x}_\mu) - \lambda^k I \right] W_k(s) \end{aligned} \tag{2.20}$$

3. Set $W_{<k+1}(s) = W_{<k}(s) + W_k(s)$ and return to step 1 until satisfied with the degree of W

We start with $k = 2$ and proceed. We elaborate on the computation of the degree k monomial $E_k(s)$ from step 1 in Section 2.4.2. Eq. (2.20) can be derived from the requirement

$$\left[F(W_{<k}(s) + W_k(s)) - (W_{<k}(\lambda s) + W_k(\lambda s)) \right]_k = 0 \tag{2.21}$$

where as before $[\cdot]_k$ denotes taking the s^k term of the quantity inside brackets. Expanding the LHS in Taylor series and discarding terms of polynomial degree greater than k gives

$$[F(W_{<k}(s)) + DF(W_{<k}(s))W_k(s) - (W_{<k}(\lambda s) + W_k(\lambda s))]_k \tag{2.22}$$

$$= E_k(s) + [DF(W_{<k}(s))W_k(s) - W_k(\lambda s)]_k \tag{2.23}$$

$$= E_k(s) + DF(\mathbf{x}_\mu)W_k(s) - W_k(\lambda s) = 0 \tag{2.24}$$

where the last line follows from the preceding one because one can divide s^k out from $E_k(s)$, $W_k(s)$, and $W_k(\lambda s)$, and then take $s \to 0$.

2.4.2 Computing $E_k(s)$: Automatic Differentiation and Jet Transport

In step 1 of the parameterization method algorithm, we computed the quantity

$$E_k(s) = [F(W_{<k}(s)) - W_{<k}(\lambda s)]_k \tag{2.25}$$

$W_{<k}(s)$ is a degree $k-1$ polynomial and λ is a constant, so the degree k term of $W_{<k}(\lambda s)$ is just 0. However, F is the nonlinear time-$T_{sc,\mu}$ mapping of phase space points by the equations of motion 1.1 and 1.2; hence, computing $F(W_{<k}(s))$ as a polynomial is not a trivial matter. For this, the tools of automatic differentiation [12] and jet transport [26] are useful.

Automatic differentiation is a technique which allows for rapid recursive evaluation of operations on polynomials. For instance, given two polynomials $f(x)$ and $g(x)$, suppose we wish to compute $d(x) = f(x)/g(x)$ as a polynomial. We know that $d(x) = f(x)/g(x) \iff f(x) = d(x)g(x)$; hence, using subscript j to denote the degree j coefficient,

$$\begin{aligned} f_k(x) &= \sum_{j=0}^{k} d_j(x) g_{k-j}(x) \\ &= \sum_{j=0}^{k-1} d_j(x) g_{k-j}(x) + d_k(x) g_0(x) \end{aligned} \tag{2.26}$$

$$\therefore d_k(x) = \frac{1}{g_0}\left(f_k(x) - \sum_{j=0}^{k-1} d_j(x) g_{k-j}(x) \right) \tag{2.27}$$

We know that $d_0 = f_0/g_0$, and using Eq. (2.27) with the known coefficients of f and g, can recursively find $d_k(x)$, $k \geq 1$. Similar recursive formulas exist for $f(x)^\alpha$ as well as many other functions [12]. The key property of all automatic differentiation formulas is that the s^k coefficient of the output depends only on the s^k and lower order coefficients of the operands. Hence, truncation of Taylor series for the purpose of implementation on a computer does not affect the accuracy of the computed coefficients.

The utility of automatic differentiation is that it allows us to substitute polynomials such as $W_{<k}(s)$ for $(x, y, \dot{x}, \dot{y})$ in the equations of motion 1.1 and 1.2 to get polynomials in s for $(\dot{x}, \dot{y}, \ddot{x}, \ddot{y})$. In particular, let $V(s,t) = \sum_{i=0}^{\infty} V_i(t) s^i : \mathbb{R}^2 \to \mathbb{R}^4$ be a Taylor series-valued function of time, with time-varying coefficients $V_i(t)$. Denote the x, y, $\dot{x}$, and $\dot{y}$ components of $V(s,t)$ as $V_x(s,t)$, $V_y(s,t)$, $V_{\dot{x}}(s,t)$, and $V_{\dot{y}}(s,t)$. Substituting V in the equations of motion, we get the system of differential equations

$$\frac{d}{dt} V_x(s,t) = V_{\dot{x}}(s,t) \tag{2.28}$$

$$\frac{d}{dt} V_y(s,t) = V_{\dot{y}}(s,t) \tag{2.29}$$

$$\frac{d}{dt} V_{\dot{x}}(s,t) = 2V_{\dot{y}}(s,t) + V_x(s,t) - (1-\mu)\frac{V_x(s,t)+\mu}{r_1(s,t)^3} - \mu\frac{V_x(s,t)-1+\mu}{r_2(s,t)^3} \tag{2.30}$$

$$\frac{d}{dt}V_{\dot{y}}(s,t) = -2V_{\dot{x}}(s,t) + V_y(s,t) - (1-\mu)\frac{V_y(s,t)}{r_1(s,t)^3} - \mu\frac{V_y(s,t)}{r_2(s,t)^3} \tag{2.31}$$

where $r_1(s,t) = \sqrt{(V_x+\mu)^2 + V_y^2}$ and $r_2(s,t) = \sqrt{(V_x - 1 + \mu)^2 + V_y^2}$. For a given t, the RHS of each equation can be simplified to a polynomial using automatic differentiation. Hence, this can be interpreted as a differential equation for the polynomial coefficients of each component of $V(s,t)$; for each equation, one simply sets the time derivative of the s^m coefficient on the LHS to the s^m coefficient on the RHS. Solving this equation with initial condition $V(s,0) = W_{<k}(s)$, we have that $V(s, T_{sc,\mu}) = F(W_{<k}(s))$, which is the polynomial we need.

Hence, by treating the coefficients of $W_{<k}(s)$ as real parameters to be integrated from 0 to $T_{sc,\mu}$, we can numerically integrate $W_{<k}(s)$ coefficient by coefficient to find $F(W_{<k}(s))$. This method of integrating a polynomial curve is known as jet transport; for more details, see Perez-Palau [26]. The essential idea is to overload algebraic operations and numerical integration routines with the ability to accept arrays of polynomial coefficients rather than only floating point numbers. We can use truncated Taylor series in this algorithm since the automatic differentiation formulas used for the evaluation of time derivatives are valid for truncated series. Note that if we have an n-dimensional state ($n = 4$ in our case) and a degree-d truncated series, then the integration required is $n(d+1)$ dimensional.

2.4.3 Notes About Computation of Manifolds

The parameterization method, automatic differentiation, and jet transport described in the preceding sections were implemented in programs written in C using the GSL library [27] for the computation of stable and unstable manifolds. Fig. 2.3 gives an example of part of the program output; in the order k step of the program, first $E_k(s) = F(W_{<k}(s)) - W_{<k}(\lambda s)$ is computed using the GSL rk8pd integrator for jet transport (denoted RK in Fig. 2.3). Printing $E_k(s)$ to the terminal, we see that the coefficients of order less than k are zero as expected in each step. The final d degree polynomial $W_{\leq d}(s)$ satisfies $F(W_{\leq d}(s)) -$

```
----------------------------------------------------------------------------
Order 2 started
RK starting for order 2...RK done for order 2

Error polynomial R(s) = F(W(s)) - W(lambda*s) =

0.000000+-0.000000*s^1+-14.018072*s^2

0.000000+-0.000000*s^1+-350.343947*s^2

-0.000000+-0.000000*s^1+-119.177120*s^2

-0.000000+0.000000*s^1+9.558964*s^2

----------------------------------------------------------------------------
Order 3 started
RK starting for order 3...RK done for order 3

Error polynomial R(s) = F(W(s)) - W(lambda*s) =

0.000000+-0.000000*s^1+0.000000*s^2+33.905013*s^3

0.000000+-0.000000*s^1+0.000000*s^2+783.828070*s^3

-0.000000+-0.000000*s^1+0.000000*s^2+347.191371*s^3

-0.000000+0.000000*s^1+-0.000000*s^2+-28.726170*s^3
```

Figure 2.3: Program Output

$W_{\leq d}(\lambda s) = 0$ up to polynomial terms of order d.

To optimize computational time and storage requirements, at the order k step, we only store polynomial coefficients up to degree k in the automatic differentiation and jet transport steps. This allows the jet transport to run much more quickly than it did when we stored additional unnecessary terms. Also, note that if $W(s)$ solves Eq. (1.8), then so does $W(\alpha s)$ where α is an arbitrary constant. Hence, if the jet transport integration is struggling to converge due to fast-growing coefficients of $W(s)$, it helps to scale $W(s)$ to $W(\alpha s)$ by multiplying the eigenvector $W_1(s)$ by $\alpha < 1$ and then restarting the parameterization method algorithm from Section 2.4.1.

Finally, one last remark is that if one takes the original periodic point $\mathbf{x}_\mu$ to be on the hyperplane $y = 0$, then using the time-reversal symmetry mentioned in Section 1.1.1, we can see that the unstable manifold $W^u(s)$ can be found from the stable manifold $W^s(s)$ simply by setting $W^u(s) = W^s(s)$ and then multiplying the y and $\dot{x}$ components of $W^u(s)$ by -1. This enables us to save half of the computation time that computing both W^s and W^u would have taken. Henceforth, we always take $\mathbf{x}_\mu$ on $y = 0$, and always use this symmetry to compute the unstable manifolds.

2.4.4 Fundamental Domains of Parameterizations

Though the d degree polynomial parameterizations $W_{\leq d}(s)$ of the stable and unstable manifolds of $\mathbf{x}_\mu$ are expected to be much more accurate than their linear approximations, they are still inexact and subject to some error. In addition, even if the polynomials could be fully and exactly computed, they still will only be valid within some radius of convergence. Hence, one must determine for which values of $s \in \mathbb{R}$ the polynomial $W_{\leq d}(s)$ is an accurate representation of the invariant manifold.

To do this, we fix an error tolerance, such as say $E_{tol} = 10^{-5}$ or 10^{-6}. We then seek to find what is referred to as the fundamental domain of $W_{\leq d}(s)$. The fundamental domain is defined as the maximum magnitude of s such that the error in invariance Eq. (1.8) is less than E_{tol}. To be precise, we want a $D \in \mathbb{R}$ such that for all s such that $|s| \leq D$,

$$\|F(W_{\leq d}(s)) - W_{\leq d}(\lambda s)\| < E_{tol} \tag{2.32}$$

By computing the fundamental domains for over 60 resonant orbit stable manifolds, we observed orders of magnitude improvement in fundamental domains for $d = 25$ compared to $d = 1$. For linear parameterizations ($d = 1$), the domains of all test cases were on the order of 10^{-4} at best, generally 10^{-5}. However, for the degree-25 polynomial parameterizations $W_{d\leq 25}(s)$, most domains were on the order of 0.1 or even 1.

Note that if one scales $W_{d\leq 25}(s)$ to $W_{\leq 25}(\alpha s)$ with $\alpha < 1$, then the fundamental domain increases by a factor of α^{-1}. Hence, whenever we compare domains between parameterizations, we always multiply the domain by any scale factor α used, so that valid comparisons can be made.

2.4.5 Globalization and Visualization

With the fundamental domains computed, we now seek to use the manifold parameterization $W(s)$ to find heteroclinic connections between different resonant periodic orbits. Before we can accomplish this, it is useful to plot the intersection of the periodic orbits' invariant manifolds with a Poincaré section. Additionally, we need to compute the manifold $W(s)$ for s values outside the fundamental domain, referred to as globalization. We do these two tasks simultaneously.

In our case, the Poincaré section we use is a $y = 0$, $x < 0$ section; recall that the Jacobi constant C is a constant of motion, so fixing the values of C and y restricts us to a 2-D surface of section. We know that the curve $W(s)$ computed earlier is an invariant manifold for the F-fixed point $\mathbf{x}_\mu$. The curve $W(s)$ lies on the 2-D invariant manifold of the resonant periodic orbit passing through $\mathbf{x}_\mu$. Hence, if we seek to find the intersection of this 2-D invariant manifold with the 2-D surface of section in the 3-D energy level submanifold, this will be a 1-dimensional curve which can be found by propagating points from the curve $W(s)$ to the section. As we are taking $\mathbf{x}_\mu$ to be on $y = 0$, only a short forwards or backwards integration should be required at each point $W(s)$.

Denote the point found by propagating $W(s)$ to the surface of section as $W_p(s)$. Henceforth, denote the forwards and backwards Poincaré maps by P_+ and P_-, respectively. Since $F(W(s)) = W(\lambda s)$ (at least within E_{tol}), we have that $P_+(W_p(s)) = W_p(\lambda s)$, and that $W_p(s)$ is a curve representing the invariant manifold for the fixed point $\mathbf{x}_\mu$ under P_+. In practice, we take a discrete grid of s-values $\{s_i\}$ from $-D$ to D (the fundamental domain value), and compute and store $W_p(s_i)$ for each s_i. For each $W_p(s_i)$, we plot the values $(x, \dot{x})$ since given C and $y = 0$, this is sufficient to determine $\dot{y}$.

Note that we no longer have a polynomial representing the manifold on the Poincaré section. Instead, we have an accurate grid of points of the manifold $W_p(s)$. Computing the polynomial representation of the manifold $W_p(s)$ on the Poincaré section requires expansion of each coefficient of $W(s)$ as a Taylor series in time under jet transport, followed by the computation of the Poincaré return time as a polynomial in s and the composition of the two polynomials, as is described by Perez-Palau [28]. Rather than carrying out this complicated procedure, we chose to simply propagate a fine grid of points to the section.

Next, we compute the manifold $W_p(s)$ for s values outside the fundamental domain. For this, we now follow the usual process of globalization of invariant manifolds, which is to propagate the fundamental domain [12]. Namely, we take the points $W_p(s_i)$, and propagate them to define $W_p(s)$ at larger s-values using the equations

$$W_p(\lambda s) = P_+(W_p(s)) \text{ if } \lambda > 1 \tag{2.33}$$

$$W_p(s/\lambda) = P_-(W_p(s)) \text{ if } \lambda < 1 \tag{2.34}$$

We then store the points of $W_p(s)$ found and their corresponding s-values in a data file. In practice, it is helpful to only count intersections such that $\dot{y}$ has the same sign as $\dot{y}$ at $\mathbf{x}_\mu$.

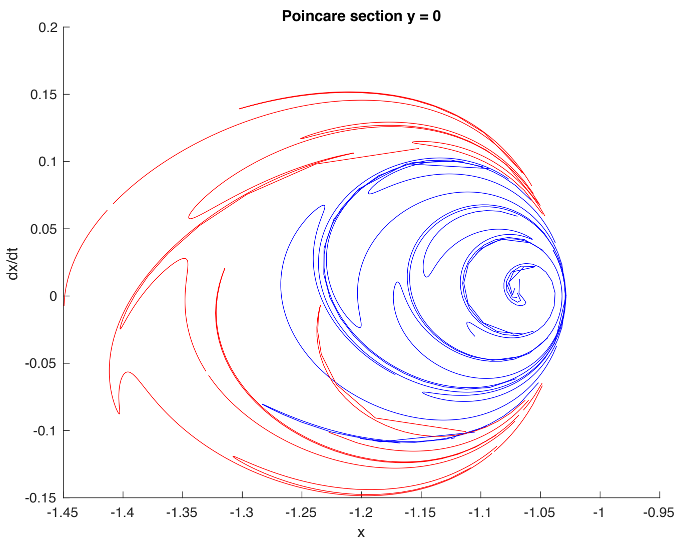


Figure 2.4: 3:4 W^u (red) and 5:6 W^s (blue) Poincaré Section for Jacobi Constant $C = 3.0024$

An example Poincaré section after globalization, with stable and unstable manifolds of 5:6 and 3:4 resonant orbits, respectively, is given in Fig. 2.4.

2.5 Computation of Heteroclinic Connections

With the stable and unstable manifolds of the PCRTBP resonant periodic orbits accurately parametrized, globalized, and plotted on the Poincaré section, we now demonstrate how to use the results of the previous computations to find heteroclinic connections between orbits. From now on, denote $W_1^u(s_u)$ and $W_2^s(s_s)$ as the intersections with the Poincaré section of the stable and unstable manifolds of periodic orbits 1 and 2, respectively. Heteroclinic connections from orbit 1 to orbit 2 correspond to intersections of the curves W_1^u and W_2^s.

We have the values of $W_1^u(s_u)$ and $W_2^s(s_s)$ on the Poincaré section on a discrete grid of s_u and s_s values, say $\{s_{u,i}\}$ and $\{s_{s,j}\}$. $W_1^u(s_u)$ and $W_2^s(s_s)$ are hence stored as sequences of consecutive points $\{W_1^u(s_{u,i})\}$ and $\{W_2^s(s_{s,j})\}$ whose $(x, \dot{x})$ values are plotted in the Poincaré section. We seek to find s_u and s_s such that $W_1^u(s_u) = W_2^s(s_s)$. To accomplish this numerically, the first part of the algorithm is to:

1. Connect all consecutive $(x, \dot{x})$ points $W_1^u(s_{u,i})$ and $W_1^u(s_{u,i+1})$ by line segments

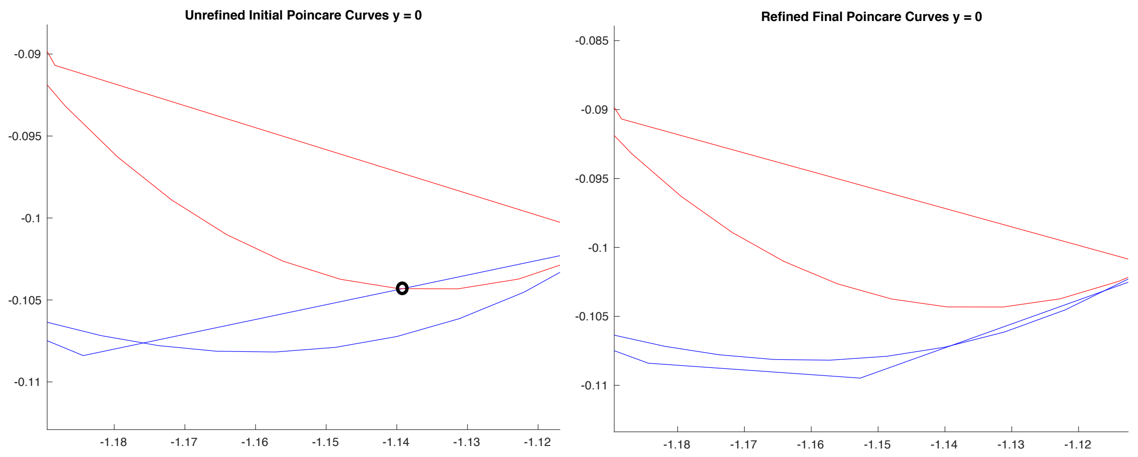


Figure 2.5: False intersection (circled) removed upon refinement

(similarly for all $W_2^s(s_{s,j})$ and $W_2^s(s_{s,j+1})$)

2. Remove all line segments corresponding to discontinuities.

3. For each segment between points of W_1^u check for intersections with all segments of W_2^s

Step 2 is somewhat heuristic; to detect discontinuities, we checked if the quantity $W_1^u(s_{u,i+1}) - W_1^u(s_{u,i})$ had large values, or if it was much larger in magnitude than $W_1^u(s_{u,i}) - W_1^u(s_{u,i-1})$ (similar for W_2^s). Also note that step 3 is easily parallelizable, and indeed benefits significantly from doing so.

With the first part of the algorithm serving to find intersecting segments of points from W_1^u and W_2^s, as well as the s_u and s_s values corresponding to the endpoints, the next part of the algorithm refines the estimate for s_u and s_s satisfying $W_1^u(s_u) = W_2^s(s_s)$. In particular, if an intersection is detected between the $\{W_1^u(a_1), W_1^u(b_1)\}$ segment and $\{W_2^s(a_2), W_2^s(b_2)\}$ segment:

1. Find $W_1^u(\frac{a_1+b_1}{2}) = P_+^k(W_1^u(\lambda_u^{-k}\frac{a_1+b_1}{2}))$ where k is such that $\lambda_u^{-k}\frac{a_1+b_1}{2}$ is in the fundamental domain of the polynomial from which W_1^u was computed

2. Find $W_2^s(\frac{a_2+b_2}{2}) = P_-^m(W_2^s(\lambda_s^{m}\frac{a_2+b_2}{2}))$ where m is such that $\lambda_s^{m}\frac{a_2+b_2}{2}$ is in the fundamental domain of the polynomial from which W_2^s was computed

3. Form the segments $\{W_1^u(a_1), W_1^u(\frac{a_1+b_1}{2})\}$, $\{W_1^u(\frac{a_1+b_1}{2}), W_1^u(b_1)\}$ and $\{W_2^s(a_2), W_2^s(\frac{a_2+b_2}{2})\}$, $\{W_2^s(\frac{a_2+b_2}{2}), W_2^s(b_2)\}$

4. Check for intersections between new segments. If found, return to step 1 with new segment endpoints replacing old ones.

5. If no intersection found, check for intersections between new segments and other segments on the same continuous curves in W_1^u and W_2^s. If found, return to step 1.

6. End bisection when $|a_1 - b_1|$ and $|a_2 - b_2|$ are small enough.

In steps 1 and 2, recall that $W_1^u(\lambda_u^{-k}\frac{a_1+b_1}{2})$ and $W_2^s(\lambda_s^m\frac{a_2+b_2}{2})$ are not given directly by the polynomials computed using the parameterization method; however, they are found by integrating points from the polynomials a short distance to the surface of section. Step 5 is necessary because sometimes, when the segments are refined into two segments, intersections that previously existed can break. An example of how this can occur is shown in Fig. 2.5, where the manifolds shown are the same $C = 3.0024$ 3:4 W^u and 5:6 W^s from Fig. 2.4.

2.6 Example Application to Resonance Transfer in the Jupiter-Europa System

The methodology described in previous sections is general, and can be applied to systems with a variety of mass ratios μ. In particular, we successfully applied the parameterization method, automatic differentiation, and jet transport to the computation of Taylor series expansions of manifolds in both the Earth-Moon and Jupiter-Europa PCRTBP systems. For the computation of heteroclinic connections, however, we focused our efforts on the Jupiter-Europa system due to the variety of missions currently being planned for that system, such as Europa Clipper [29], Europa Lander [20], and Jupiter Icy Moons Explorer [30].

We used the tools developed in the previous sections for the computation of a 3:4 to 5:6 resonance transfer trajectory in the PCRTBP at the Jacobi constant value 3.0024. The

Table 2.1: Initial conditions, periods, and eigenvalues for 3:4 and 5:6 resonant periodic orbits at $C = 3.0024$.

	$5:6$	$3:4$
x	-1.231240907544348	-1.391929713356257
y	0.000000000000000	1.4178538082815e-18
$\dot{x}$	0.000000000000000	-2.9260154691618e-14
$\dot{y}$	0.371411618064504	0.609863420586548
T_{sc}	38.328135171743014	25.338526603095760
λ_s	0.001256465177783	0.011341070996024
λ_u	795.8835769446018	88.175093899915780

initial conditions, periods T_{sc}, and monodromy matrix eigenvalues corresponding to each periodic orbit are given in Table 2.1.

Using the parameterization method described in Section 2.4.1, we obtained degree 50 Taylor polynomial expansions representing the stable manifolds of the points in Table 2.1 under the time T_{sc} map by the equations of motion. By the PCRTBP time-reversal symmetry, we also obtain the unstable manifolds. Next, upon computation of the fundamental domains of these polynomials (using $E_{tol} = 10^{-5}$), we found that the domain for the 5:6 orbit polynomial was approximately 0.9904, while that for the 3:4 orbit was approximately 0.7146. The globalization and Poincaré section visualization routine described in Section 2.4.5 was then applied to the computed polynomials. Globalization is necessary as the manifold parameterizations, when propagated to the Poincaré section, do not intersect within the fundamental domain values of the parameters.

As before, we denote $W^u_{3:4}$ and $W^s_{5:6}$ as being the unstable and stable manifolds of the Poincaré map fixed points corresponding to the 3:4 and 5:6 orbit points from Table 2.1. The computed Poincaré section with $W^u_{3:4}$ and $W^s_{5:6}$ was shown earlier in Fig. 2.4. With the Poincaré section points computed and stored for both $W^u_{3:4}$ and $W^s_{5:6}$, we then proceeded to compute heteroclinic connections using the bisection method described in Section 2.5.

6 intersections between segments of consecutive stored $W^u_{3:4}$ and $W^s_{5:6}$ points were initially detected; however, upon refining the segments through the algorithm from Section 2.5, 3 preliminary intersections were found to be spurious. The coordinates of the 3 com-

Table 2.2: Computed Heteroclinic Connection Points and corresponding s_s, s_u Values

	1	2	3
x	-1.2265598	-1.2230160	-1.1110838
y	-4.101840e-14	-1.989706e-14	5.780044e-15
$\dot{x}$	-0.060806259	-0.063340619	-0.10187786
$\dot{y}$	0.35908692	0.35309042	0.14762036
s_s	-301.609248	-295.877551	14.24735921
s_u	-3785.98948	-3706.35853	-3874.28227

puted actual connections are given in Table 2.2. Fig. 2.6 shows how the program refined the Poincaré section in the neighborhood of each computed intersection in order to precisely compute the heteroclinic connection point. Finally, Fig. 2.7 shows the trajectory corresponding to the third heteroclinic connection point from Table 2.2, with the start 3:4 periodic orbit shown in red and the destination 5:6 periodic orbit shown in blue.

Note that our approach of using high order parameterizations of invariant manifolds to compute heteroclinic connections bears some similarity with prior studies; for instance, James and Murray [31] parameterized manifolds of periodic orbits using high order Chebyeshev-Taylor series, using the resulting 2D parameterizations to find connecting orbits. However, our study avoids dealing with 2D manifolds by using a Poincaré section to reduce the dimensionality of the problem, without sacrificing the accuracy which comes from using high order manifold expansions.

2.7 Conclusions

In this chapter, we studied the persistence of resonant periodic orbits in the PCRTBP, and subsequently demonstrated the application of the parameterization method for the computation of high-order expansions of resonant orbit invariant manifolds. We also then demonstrated how to use the resulting polynomials to calculate useful heteroclinic connections. We were able to develop tools to find polynomial approximations of resonant orbit stable and unstable manifolds of degree 25 or even higher; these expansions resulted in a 1000x improvement in the domains of accuracy of the manifold representations as compared to

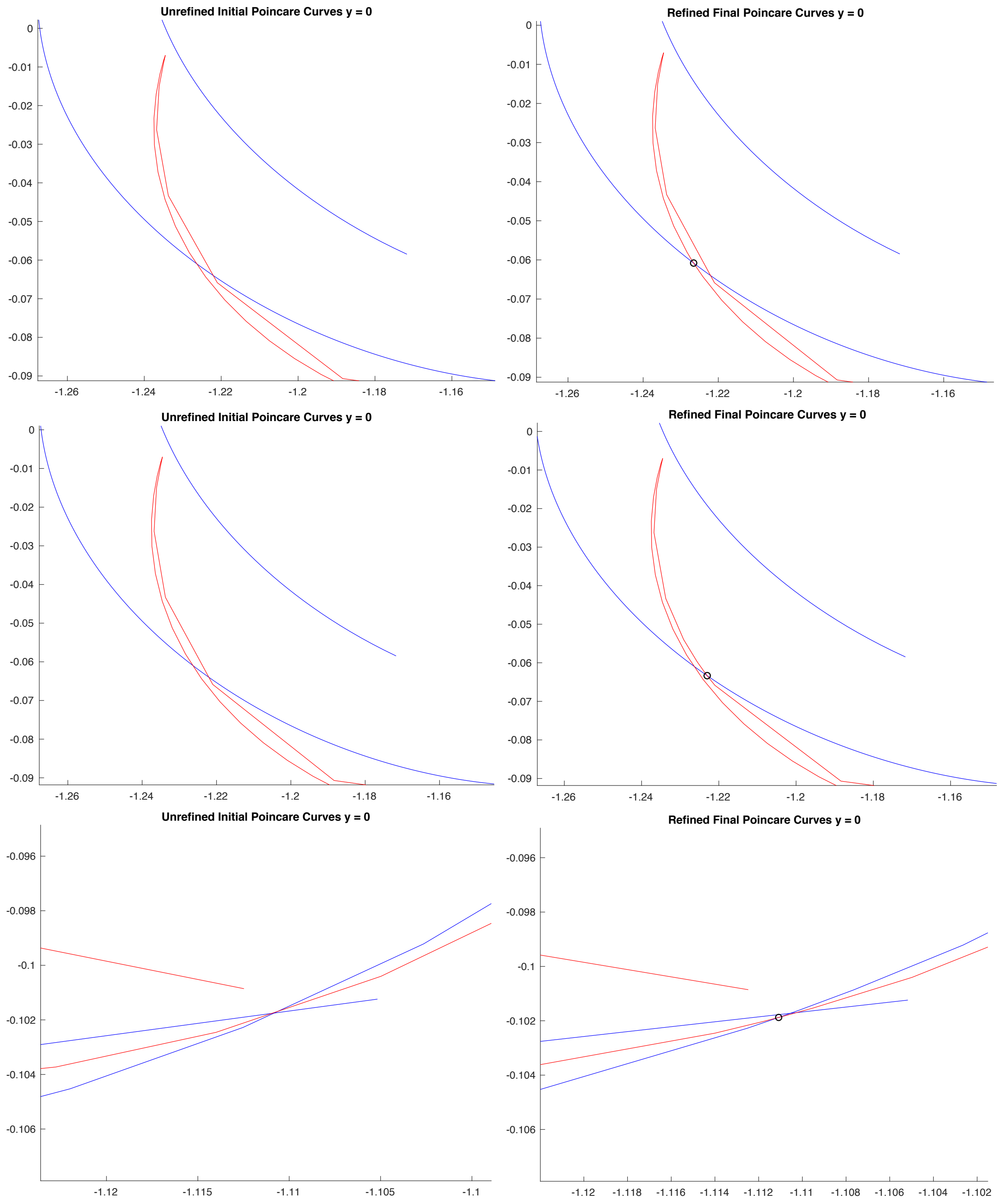


Figure 2.6: Examples of Approximate Intersections (Left) and Computed Heteroclinic Connections (Right, Circled) for 3:4 to 5:6 Resonance Transfer at Jacobi Constant $C = 3.0024$ ($W^u_{3:4}$ red and $W^s_{5:6}$ blue)

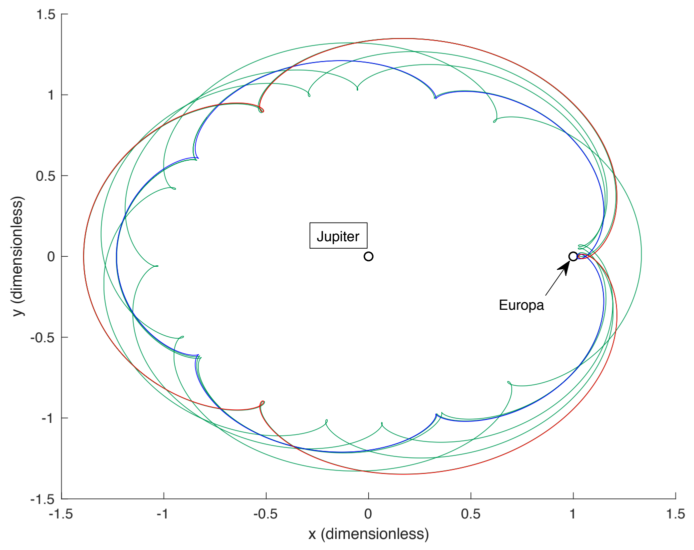


Figure 2.7: Trajectory Corresponding to Heteroclinic Connection 3 from Table 2.2

just using linear approximations.

The tools developed were tested in the Jupiter-Europa system, with the calculations of the manifolds and connections taking only a few minutes on a laptop for a given pair of resonances. The manifold polynomials were used to successfully compute several connections corresponding to 3:4 to 5:6 resonance transition, demonstrating the usefulness of these parameterizations for mission design.

CHAPTER 3
RAPID AND ACCURATE METHODS FOR COMPUTING WHISKERED TORI AND THEIR MANIFOLDS IN PERIODICALLY PERTURBED PCRTBP MODELS

3.1 Introduction

Numerous studies have been carried out in recent years where quasi-periodic orbits of various restricted 3 or 4-body models have been computed and used for applications to space mission design. For instance, [32] studied periodic and quasi-periodic orbits in the phase space of the Augmented Hill 3-Body problem near the L_1 and L_2 libration points. [33] applied collocation methods to the computation of invariant tori near L_1 and L_2 in both spatial circular restricted 3-body problem (CRTBP) as well as periodically-perturbed planar CRTBP models. And looking further in the past, the book series of [34] presented many other computational methods and applications for quasi-periodic orbits near libration points. However, all of these studies, as well as almost all other prior research, use methods of computing tori which require solving large-dimensional linear systems of equations at each step of the differential correction. Furthermore, while the quasi-periodic orbits are computed successfully using those methods, stability information including stable and unstable directions must be computed separately. Also, in most prior work, these linear stable/unstable directions are directly used as approximate local stable/unstable manifolds for the tori, neglecting higher order terms and thus losing accuracy.

Another characteristic of the vast majority of prior research, including the previously mentioned studies, is that the analysis focuses on tori associated with the libration points, such as Lissajous or quasi-halo orbits. [35] did compute invariant tori near stable resonant periodic orbits in the planar circular restricted 3-body problem (PCRTBP), but the tori computed are stable, without stable or unstable manifolds. Unstable resonant periodic orbits and their stable and unstable manifolds are known to be important mechanisms of dynamical transport in the interior and exterior realms of the CRTBP [36], and these orbits have seen significant interest and use as a tool for trajectory design in multi-body systems. For example, out of the nine Titan-to-Titan encounters made by Cassini between July 2013 and June 2014, eight of the nine resulting transfers involved resonances [19]. More recently, the baseline mission design for the Europa Lander mission concept made profitable use of these mechanisms for the final approach to the surface of Europa [20]. For further examples and more background on resonant orbits, see [21].

In this study, we develop efficient algorithms which enable simultaneous computation of not only unstable invariant tori, but also of their center, stable, and unstable bundles (directions) in periodically perturbed PCRTBP models. Solving for bundles alongside the tori actually allows us to avoid solving large linear systems, thus improving the algorithmic efficiency of our method compared to tori-only methods used in previous investigations. We apply our tools to the computation of unstable tori and bundles near PCRTBP resonances, using the Jupiter-Europa planar elliptic RTBP as the dynamical model for demonstration and a solution tolerance of 10^{-7}. Next, we use the results of the preceding step to start a recursive parameterization method [23, 37, 12] for the computation of high order Fourier-Taylor approximations of the stable and unstable manifolds of the tori. We demonstrate improvements in manifold accuracy as compared with the linear manifold approximations used in other studies; these parameterizations can also be differentiated, which is useful for computing intersections of manifolds. Finally, we develop a Levi-Civita regularization for the equations of motion, which is used to globalize the parameterized manifolds even when they pass through singularities of the equations of motion.

3.2 Resonant Periodic Orbits

Before starting on the development of our algorithms, we reiterate some properties of resonant orbits, discussed in Chapter 2, which are relevant for the discussion to follow. Mean-motion resonances are PCRTBP periodic orbits which, by definition, persist from elliptical orbits of the Kepler problem, and hence are not in the center manifold of any of the libration points. Their main characteristic is that their orbital periods are nearly rational multiples of 2π, the period of m_1-m_2 motion (the periods become exact rational multiples of 2π as $\mu \to 0$). A family of resonant periodic orbits is characterized by a ratio $m : n$, $m, n \in \mathbb{Z}^+$. This notation means that in an inertial reference frame, the spacecraft makes approximately m revolutions about m_1 in the time that m_1 and m_2 revolve n times around their barycenter.

For a given resonance $m : n$ in the PCRTBP with $\mu > 0$, there typically exist one stable and one unstable resonant periodic orbit inside the submanifold $H_0(x, y, p_x, p_y) = E$, for each fixed value of E in some interval of energy values $[E_{min}, E_{max}]$ [13]. This gives us continuous families of stable and unstable resonant periodic orbits; the periods of the orbits within a given family vary with E. The unstable resonant periodic orbits have monodromy (Floquet) matrix eigenvalues 1, 1, λ_s, and $\lambda_u = \lambda_s^{-1}$, where $|\lambda_s| < 1$. Thus, there are 2D stable and unstable manifolds attached to the unstable resonant periodic orbits. These manifolds serve as low-energy pathways to and from these periodic orbits. Furthermore, the manifolds of different resonances at the same energy level can intersect in the PCRTBP, giving heteroclinic connections which allow for propellant-free resonance transitions.

3.3 Normally Hyperbolic Invariant Manifolds and Existence of Tori

As discussed in Section 3.2, for each value of E in some interval of energy values $[E_{min}, E_{max}]$, there exists one unstable $m : n$ resonant periodic orbit in the PCRTBP. Each of these periodic orbits is diffeomorphic to a circle $\mathbb{T}$. Now, consider the union of all the unstable $m : n$ resonant periodic orbits for all values of $E \in [E_{min}, E_{max}]$. The resulting set is a 2D manifold Ξ diffeomorphic to $\mathbb{T} \times [0,1]$ in the 4D PCRTBP phase space. Furthermore, at each point of Ξ, there are stable and unstable directions transverse to the manifold, which come from the stable and unstable eigenvectors of the periodic orbits which make up this manifold. Since the phase space is 4D and Ξ is 2D, at any point of Ξ, the stable and unstable directions together with the 2D manifold tangent space span the entire phase space. This means that Ξ is a normally hyperbolic invariant manifold (NHIM) of the PCRTBP flow; in fact, any family of unstable PCRTBP periodic orbits forms such a NHIM. For a rigorous definition of NHIMs for flows, see [38].

NHIMs are important because they persist under sufficiently small perturbations of the equations of motion [38]; as our numerical results later in this chapter will demonstrate, the perturbations we study are indeed "sufficiently small". However, to apply this NHIM persistence result to our case of time-periodic perturbations, the original and perturbed systems must be defined on the same phase space. Hence, we take the PCRTBP from its original 4D phase space (x, y, p_x, p_y) to the 5D extended phase space $(x, y, p_x, p_y, \theta_p)$, $\theta_p \in \mathbb{T}$. We define $\dot{\theta}_p = \Omega_p$ for the unperturbed PCRTBP in extended phase space, while x, y, p_x, and p_y still follow Equations (1.3) and (1.4). Hence, periodic orbits of period T_1 from the original 4D PCRTBP phase space become 2D quasi-periodic orbits in the PCRTBP defined on the extended phase space, with one of the frequencies being $\Omega_1 = 2\pi/T_1$ and the other being Ω_p, unless Ω_1/Ω_p is rational. The NHIM Ξ from the original phase space becomes the NHIM $\bar{\Xi} = \Xi \times \mathbb{T}$ in the extended phase space due to the extra angle. Hence, $\bar{\Xi}$ is diffeomorphic to $\mathbb{T}^2 \times [0,1]$.

Now, since the PCRTBP and its NHIM have been transferred to the same extended phase space as the periodically-perturbed models, we can conclude that for $\varepsilon > 0$ sufficiently small, $\bar{\Xi}$ will persist as a NHIM $\bar{\Xi}_\varepsilon$ of the perturbed equations of motion (1.5) and (1.6). $\bar{\Xi}_\varepsilon$ will be diffeomorphic to $\bar{\Xi}$ and hence also to $\mathbb{T}^2 \times [0,1]$. Furthermore, note that $\bar{\Xi}$ in the extended phase space PCRTBP is foliated by 2D invariant tori, since Ξ was foliated by periodic orbits. From KAM theory [39], we can expect that inside $\bar{\Xi}_\varepsilon$, the invariant tori from $\bar{\Xi}$ with sufficiently non-resonant frequencies Ω_1 and Ω_p will also persist after perturbation with only small gaps between them. Hence, we will focus this study on these 2D tori in the periodically-perturbed PCRTBP models.

3.3.1 Stroboscopic Maps, NHIMs, and Invariant Circles

Any 2D invariant torus in the periodically-perturbed PCRTBP extended phase space can be parameterized as the image of a function of two angles $K_2 : \mathbb{T}^2 \to \mathbb{R}^4 \times \mathbb{T}$. A quasi-periodic trajectory $\mathbf{x}(t)$ lying on this torus can be expressed as

$$\mathbf{x}(t) = K_2(\theta, \theta_p) \qquad \theta = \theta_0 + \Omega_1 t, \quad \theta_p = \theta_{p,0} + \Omega_p t \tag{3.1}$$

where θ_0 and $\theta_{p,0}$ are determined from the initial condition $\mathbf{x}(0)$, and $\Omega_1 = 2\pi/T_1$, where T_1 is the period of the PCRTBP periodic orbit associated with the torus. θ_p is the same perturbation phase angle defined in Section 1.1.2, so one of the two torus frequencies will be Ω_p. We can then define the stroboscopic map $F_\varepsilon : \mathbb{R}^4 \times \mathbb{T} \to \mathbb{R}^4 \times \mathbb{T}$ as the time-$2\pi/\Omega_p$ mapping of extended phase space points by the equations of motion (1.5) and (1.6) with perturbation parameter ε. We find that

$$F_\varepsilon(K_2(\theta, \theta_p)) = K_2(\theta + \omega, \theta_p), \text{ where } \omega = 2\pi\Omega_1/\Omega_p \tag{3.2}$$

since the angle θ_p advances by 2π in the time $2\pi/\Omega_p$ and is therefore invariant under F_ε. Hence, we can fix θ_p (for our PERTBP test case we choose $\theta_p = 0$), and then define $K(\theta) = K_2(\theta, \theta_p)$. Then, Eq. (3.2) becomes

$$F_\varepsilon(K(\theta)) = K(\theta + \omega) \tag{3.3}$$

By ignoring the last fixed θ_p component of the extended phase space and a slight abuse of notation, we can consider $F_\varepsilon : \mathbb{R}^4 \to \mathbb{R}^4$ and $K : \mathbb{T} \to \mathbb{R}^4$.

Eq. (3.3) implies that K parameterizes an invariant 1D torus of the map F_ε. It is significantly more computationally efficient to compute 1D invariant tori (invariant circles) K of the map F_ε in 4D phase space than 2D tori K_2 of the flow in the 5D extended phase space. The reason for this is that the reduction in the dimension of the torus helps mitigate the curse of dimensionality (see remark 3.3.1). Hence, from this point onwards, we will consider the map F_ε and its invariant circles and manifolds, rather than invariant objects of the continuous time flow. Similar approaches are also used by [40] and [41]. Note that the computation of F_ε is just the integration of an ODE.

As a final note, the stroboscopic map allows us to use the theory of NHIMs of maps [38, 42] to understand the presence of invariant circles in periodically-perturbed PCRTBP models. In particular, note that unstable periodic orbits of the unperturbed PCRTBP are also unstable invariant circles of the map $F_{\varepsilon=0}$. Hence, the PCRTBP flow NHIM Ξ defined at the beginning of Section 3.3 is also a NHIM of the map $F_{\varepsilon=0}$. Just as in the case of flows, the theory shows that NHIMs of maps persist under sufficiently small perturbations of the map. Hence, for sufficiently small $\varepsilon > 0$, Ξ will persist as a NHIM Ξ_ε of F_ε, with Ξ_ε diffeomorphic to Ξ and hence also to $\mathbb{T} \times [0, 1]$. Furthermore, since Ξ is foliated by invariant circles whose rotation numbers satisfy a twist condition, KAM theory [39] tells us that that the invariant circles with sufficiently irrational rotation number ω persist inside Ξ_ε for $\varepsilon > 0$.

Remark. *The evaluation of F_ε can be computationally expensive. Hence, one may wonder if the dimension reduction actually helps the computation efficiency or not. However, the problems of propagating in time and computing the tori are numerically very different; while numerical integration remains very feasible for all the values of ε, tori can break down for larger values of the parameter. Hence, the flow torus parameterization K_2 has very anisotropic regularity and behavior. It remains extremely smooth in the flow direction, but in the transversal direction, it may lose differentiability. Using algorithms that recognize this effect is advantageous.*

Also, the problem of integrating ODE's has been extensively studied over many years and there are many efficient algorithms that can be tried, including adaptive algorithms that use smaller step sizes on small spots where the equation is stiff. However, computing the 2D torus parameterization K_2 requires a uniform grid discretizing $\mathbb{T}^2$, which would result in unnecessarily large numbers of discretization points throughout the trajectory. For our algorithm, the operation count is close to linear in the number of grid points, so the cost of adding one more dimension would be significant. Finally, also note that numerically integrating a grid of points is very readily parallelizable by assigning each trajectory to a thread.

3.4 A Parameterization Method for Computing Invariant Tori and Bundles

In this section, we develop and implement a parameterization method for the simultaneous computation of unstable invariant tori as well as their center, stable, and unstable directions, also known as bundles, for stroboscopic maps of periodically-perturbed PCRTBP models. The method works both for tori with cylindrical stable/unstable bundles as well as for those whose bundles are Möbius strips (see Section 3.4.8). We present the analytical details and derivation of the method, as well as the considerations required for its discretization and numerical implementation in a computer program. Our method is broadly inspired by those of [12], except for the additional presence of a center bundle which is not considered by them and requires extra calculations. A different but conceptually related method can also be found in the work of [43].

3.4.1 Equations for Parameterization Method for Invariant Tori and Bundles

For notational convenience, denote the stroboscopic map F_ε from Section 3.3.1 as F from now on. Assume we are computing an $\varepsilon > 0$ invariant circle corresponding to a PCRTBP periodic orbit of known period T_1; this fixes the rotation number $\omega = 2\pi\Omega_1/\Omega_p$ since $\Omega_1 = 2\pi/T_1$. As given in Eq. (3.3), we wish to find a parameterization $K : \mathbb{T} \to \mathbb{R}^4$ of the

F-invariant circle satisfying the torus invariance equation

$$F(K(\theta)) = K(\theta + \omega) \tag{3.4}$$

Eq. (3.4) is equivalent to the framework of Section 1.2 with $M = \mathbb{R}^4$, $\mathcal{M} = \mathbb{T}$, and $f(s) = s + \omega$. In addition, for our quasi-Newton method, we will add another equation to be solved for matrix-valued periodic functions $P(\theta)$, $\Lambda(\theta) : \mathbb{T} \to \mathbb{R}^{4\times 4}$ such that

$$DF(K(\theta))P(\theta) = P(\theta + \omega)\Lambda(\theta) \tag{3.5}$$

Furthermore, we mandate that $\Lambda(\theta)$ has the form

$$\Lambda(\theta) = \begin{bmatrix} 1 & T(\theta) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \lambda_s(\theta) & 0 \\ 0 & 0 & 0 & \lambda_u(\theta) \end{bmatrix} \tag{3.6}$$

for some functions $T(\theta), \lambda_s(\theta), \lambda_u(\theta) : \mathbb{T} \to \mathbb{R}$ to be solved for. The form of Eq. (3.6) is motivated by geometric considerations that we will detail in Section 3.4.2.

As will be explained at the end of Section 3.4.4, solving simultaneously for K, P, and Λ is actually more efficient than solving for K alone; the quasi-Newton method we will present for solving Eq. (3.4)-(3.5) uses the near-diagonal form of Λ to decouple the linear system of equations we get in each differential correction step. The method will require only algebraic operations, phase shifts, and the solving of 1D equations for scalar-valued functions.

3.4.2 Understanding the P and Λ Matrices

In addition to their numerical utility, P and Λ have a geometric significance which will be useful when computing stable and unstable manifolds later on. Since K is contained in the 2D normally hyperbolic invariant manifold Ξ_ε defined at the end of Section (3.3.1), we know that there are tangent, center, stable, and unstable directions to the torus at each point $K(\theta)$. The columns of P will be these four vector bundles, with $\lambda_s(\theta)$ and $\lambda_u(\theta)$ set to the stable and unstable multipliers for the corresponding bundles. To see why, consider Eq. (3.5) column by column.

Let $\mathbf{v}_t(\theta), \mathbf{v}_c(\theta)$, $\mathbf{v}_s(\theta)$, and $\mathbf{v}_u(\theta)$ denote the first, second, third, and fourth columns of $P(\theta)$, respectively. Then, Equations (3.5) and (3.6) are equivalent to

$$DF(K(\theta))\mathbf{v}_t(\theta) = \mathbf{v}_t(\theta+\omega) \tag{3.7}$$

$$DF(K(\theta))\mathbf{v}_c(\theta) = T(\theta)DK(\theta+\omega) + \mathbf{v}_c(\theta+\omega) \tag{3.8}$$

$$DF(K(\theta))\mathbf{v}_s(\theta) = \lambda_s(\theta)\mathbf{v}_s(\theta+\omega) \tag{3.9}$$

$$DF(K(\theta))\mathbf{v}_u(\theta) = \lambda_u(\theta)\mathbf{v}_u(\theta+\omega) \tag{3.10}$$

First of all, note that Eq. (3.9)-(3.10) are the definition of stable and unstable bundles $\mathbf{v}_s(\theta)$ and $\mathbf{v}_u(\theta)$ and multipliers $\lambda_s(\theta)$ and $\lambda_u(\theta)$ for the torus K. Hence, the third and fourth columns of P satisfy Eq. (3.5) if and only if they are torus stable and unstable bundles, respectively. Also, differentiating Eq. (3.4) gives

$$DF(K(\theta))DK(\theta) = DK(\theta+\omega) \tag{3.11}$$

which shows that $\mathbf{v}_t(\theta) = DK(\theta)$ solves Eq. (3.7). As a result, the first column of P can be set as the torus tangent bundle $DK(\theta)$; in fact, if Eq. (3.5) has a solution, it is easy to show that column 1 of P *must* be $\alpha DK(\theta)$ for some $\alpha \in \mathbb{R}$.

Finally, since F is a Hamiltonian flow map and hence is symplectic, given $K(\theta)$, $\mathbf{v}_s(\theta)$, and $\mathbf{v}_u(\theta)$, we can find $\mathbf{v}_c(\theta)$ solving Eq. (3.8) for some function $T : \mathbb{T} \to \mathbb{R}$; we postpone the description and proof of how to compute such a $\mathbf{v}_c(\theta)$ to Section 3.4.8 where the method will be used. Any such $\mathbf{v}_c(\theta)$ is known as a symplectic conjugate to $DK(\theta)$, and is a center direction to the torus K [44]. Hence, column 2 of P satisfies Eq. (3.5) if and only if it is a symplectic conjugate center bundle. We should note that Eq. (3.5) is actually underdetermined; symplectic conjugates are not unique, and we can change the scales of the stable and unstable bundles at each θ; we will take advantage of this in Section 3.4.7 to make Λ constant.

3.4.3 Summary of Steps for Quasi Newton-Method for Tori and Bundles

We will now develop our quasi-Newton method for solving Eq. (3.4) and (3.5). Before presenting the details of the method, we give a brief overview. Assume we have an approximate solution (K, P, Λ) for Eq. (3.4)-(3.5). Then, we will

1. Compute $E(\theta) = F(K(\theta)) - K(\theta + \omega)$, $E_{red}(\theta) = P^{-1}(\theta + \omega)DF(K(\theta))P(\theta) - \Lambda(\theta)$
2. Solve $-P^{-1}(\theta + \omega)E(\theta) = \Lambda(\theta)\xi(\theta) - \xi(\theta + \omega)$ for $\xi : \mathbb{T} \to \mathbb{R}^4$ using Eq. (3.17)-(3.20) and set $K(\theta)$ equal to $K(\theta) + P(\theta)\xi(\theta)$ (details given in Section 3.4.4).
3. Set column 1 of $P(\theta)$ to $DK(\theta)$. Recompute $DF(K(\theta))$ and $E_{red}(\theta)$.
4. Solve $-E_{red}(\theta) = \Lambda(\theta)Q(\theta) - Q(\theta + \omega)\Lambda(\theta) - \Delta\Lambda(\theta)$ for $Q : \mathbb{T} \to \mathbb{R}^{4\times 4}$ and $\Delta\Lambda$ using Eq. (3.32)-(3.43). Set $P(\theta)$ equal to $P(\theta) + P(\theta)Q(\theta)$ and $\Lambda(\theta)$ equal to $\Lambda(\theta) + \Delta\Lambda(\theta)$ (details given in Section 3.4.5).
5. Return to step 1 and repeat correction until E and E_{red} are within tolerance.

3.4.4 Quasi-Newton Step for Correcting K

We seek to solve Eq. (3.4) and (3.5) for K, P, and Λ. All the entries of Λ are fixed as 0 or 1 as shown in Eq. (3.6) except for $T(\theta)$, $\lambda_s(\theta)$, and $\lambda_u(\theta)$. We will now derive an iterative step that, given an approximate solution (K, P, Λ) of Eq. (3.4) and (3.5), produces a much more accurate one. Define the errors

$$E(\theta) = F(K(\theta)) - K(\theta + \omega) \tag{3.12}$$

$$E_{red}(\theta) = P^{-1}(\theta+\omega)DF(K(\theta))P(\theta) - \Lambda(\theta) \tag{3.13}$$

We then need to find corrections ΔK, ΔP, and $\Delta\Lambda$ to cancel E and E_{red}. We start with ΔK; write $\Delta K(\theta) = P(\theta)\xi(\theta)$. We will solve for $\xi : \mathbb{T} \to \mathbb{R}^4$ satisfying

$$\eta(\theta) \overset{\text{def}}{=} -P^{-1}(\theta+\omega)E(\theta) = \Lambda(\theta)\xi(\theta) - \xi(\theta+\omega) \tag{3.14}$$

Claim. *For ω sufficiently irrational and E and E_{red} sufficiently small, if ξ solves Eq.* (3.14)*, then adding $\Delta K = P\xi$ to K reduces the error E quadratically.*

Remark. *We use the phrase "sufficiently irrational" when describing conditions on ω that ensure the validity of our quasi-Newton method. For those aware of KAM theory, what we mean by this is that ω is Diophantine, as most numbers are [45]. This condition is useful due to the classic small-divisors problem when solving cohomological equations (see Eq.* (3.24)*).*

Proof. Substitute $K(\theta) + \Delta K(\theta)$ into the RHS of Eq. (3.12). Assuming that ΔK is small enough (true for E sufficiently small and ω sufficiently irrational), we can expand Eq. (3.12) in Taylor series to get

$$\begin{aligned} E_{new}(\theta) &= F(K(\theta) + \Delta K(\theta)) - [K(\theta+\omega) + \Delta K(\theta+\omega)] \\ &= F(K(\theta)) + DF(K(\theta))\Delta K(\theta) + \mathcal{O}(\Delta K(\theta)^2) - [K(\theta+\omega) + \Delta K(\theta+\omega)] \\ &= E(\theta) + DF(K(\theta))\Delta K(\theta) - \Delta K(\theta+\omega) + \mathcal{O}(\Delta K(\theta)^2) \end{aligned} \tag{3.15}$$

$\Delta K = P\xi$, and Eq. (3.13) implies $DF(K(\theta))P(\theta) = P(\theta+\omega)\left[\Lambda(\theta) + E_{red}(\theta)\right]$. Thus,

$$\begin{aligned} E_{new}(\theta) &= E(\theta) + DF(K(\theta))P(\theta)\xi(\theta) - P(\theta+\omega)\xi(\theta+\omega) + \mathcal{O}(\xi(\theta)^2) \\ &= E(\theta) + P(\theta+\omega)\left[\Lambda(\theta)\xi(\theta) + E_{red}(\theta)\xi(\theta) - \xi(\theta+\omega)\right] + \mathcal{O}(\xi(\theta)^2) \\ &= P(\theta+\omega)E_{red}(\theta)\xi(\theta) + \mathcal{O}(\xi(\theta)^2) \end{aligned} \tag{3.16}$$

where the last line follows from Eq. (3.14). For ω sufficiently irrational, ξ will be similar in magnitude to E, so $E_{red}(\theta)\xi(\theta)$ will be quadratically small, comparable to $E_{red}(\theta)E(\theta)$. Hence, as long as E (and hence ξ and ΔK) are small enough that the Taylor expansion in Eq. (3.15) is valid, and the $\mathcal{O}(\xi^2)$ terms of the Taylor expansion are small, the new error E_{new} will be quadratically smaller than E. $\square$ $\square$

To solve Eq. (3.14), let $\xi(\theta) = \begin{bmatrix}\xi_1 & \xi_2 & \xi_3 & \xi_4\end{bmatrix}^T$ and $\eta_(\theta) = \begin{bmatrix}\eta_1 & \eta_2 & \eta_3 & \eta_4\end{bmatrix}^T$. As Λ is nearly diagonal, we can write Eq. (3.14) component-wise as

$$\eta_1(\theta) - T(\theta)\xi_2(\theta) = \xi_1(\theta) - \xi_1(\theta+\omega) \tag{3.17}$$

$$\eta_2(\theta) = \xi_2(\theta) - \xi_2(\theta+\omega) \tag{3.18}$$

$$\eta_3(\theta) = \lambda_s(\theta)\xi_3(\theta) - \xi_3(\theta+\omega) \tag{3.19}$$

$$\eta_4(\theta) = \lambda_u(\theta)\xi_4(\theta) - \xi_4(\theta+\omega) \tag{3.20}$$

Fixed-Point Iteration: Solving for ξ_3 and ξ_4

To solve for ξ_3 and ξ_4, rewrite Equations (3.19) and (3.20) in the form

$$\xi_3(\theta) = \lambda_s(\theta-\omega)\xi_3(\theta-\omega) - \eta_3(\theta-\omega) \overset{\text{def}}{=} [A(\xi_3)](\theta) \tag{3.21}$$

$$\xi_4(\theta) = \lambda_u^{-1}(\theta)\left[\eta_4(\theta) + \xi_4(\theta+\omega)\right] \overset{\text{def}}{=} [B(\xi_4)](\theta) \tag{3.22}$$

We define A as a map from functions to functions, which sends any $f(\theta) : \mathbb{T} \to \mathbb{R}$ to the new function $[A(f)](\theta) = \lambda_s(\theta-\omega)f(\theta-\omega) - \eta_3(\theta-\omega)$; B is defined similarly using Eq. (3.22). To find ξ_3 and ξ_4, let $\xi_{3,0} = \xi_{4,0} = 0$ and iterate $\xi_{3,n+1} = A(\xi_{3,n})$ and $\xi_{4,n+1} = B(\xi_{4,n})$ repeatedly, starting at $n = 0$. The iterations will converge to the desired solutions ξ_3 and ξ_4 of Eq. (3.21) and (3.22). We now explain why.

Lemma 1. *A, B are contraction maps, hence the iterations $\xi_{3,n+1} = A(\xi_{3,n})$, $\xi_{4,n+1} = B(\xi_{4,n})$ uniformly converge exponentially fast as $n \to \infty$ to the solutions ξ_3 and ξ_4.*

Proof. Note that $|\lambda_s(\theta)| < 1$ and $|\lambda_u^{-1}(\theta)| < 1$ for all $\theta \in \mathbb{T}$. Let $f_1, f_2 : \mathbb{T} \to \mathbb{R}$ be two continuous functions, and define $C = \max_{\theta\in\mathbb{T}} |\lambda_s(\theta)| < 1$. We have that

$$\begin{aligned}
&\max_{\theta\in\mathbb{T}} \|[A(f_1)](\theta) - [A(f_2)](\theta)\| \\
&\qquad = \max_{\theta\in\mathbb{T}} \|\lambda_s(\theta-\omega)f_1(\theta-\omega) - \lambda_s(\theta-\omega)f_2(\theta-\omega)\| \\
&\qquad \leq C\max_{\theta\in\mathbb{T}} \|f_1(\theta-\omega) - f_2(\theta-\omega)\| = C\max_{\theta\in\mathbb{T}} \|f_1(\theta) - f_2(\theta)\|
\end{aligned} \tag{3.23}$$

As $C < 1$, A is a contraction map under the uniform norm; the same can be shown for B very similarly. The contraction mapping theorem [46] tells us that every such map has a unique fixed point; furthermore, the fixed point can be found by iterating any value in the domain of the map forwards until convergence. The solutions of Equations (3.21) and (3.22) are by definition the fixed points of contraction maps A and B. Hence, the iterations converge to ξ_3 and ξ_4. □ □

Remark. *If λ_s and λ_u are constant, Fourier methods (see Section 3.4.4) can also be used to solve Eq.* (3.19)-(3.20). *This is useful if $\lambda_s, \lambda_u \approx 1$. When using the quasi-Newton method for continuation, one can ensure constant λ_s, λ_u throughout the correction by applying the procedure of Section 3.4.7 to the solution K, P, Λ used for continuation initialization, and following the instructions in Remark 3.4.5 when correcting P, Λ during each quasi-Newton step. We did not ensure constant λ_s, λ_u in our algorithm implementation, and used fixed-point iteration instead.*

Cohomological Equations: Solving for ξ_1 and ξ_2

We next solve Eq. (3.18) for ξ_2, which is then used in the LHS of Eq. (3.17) to solve for ξ_1. In both cases, we must solve cohomological equations of form

$$b(\theta) = a(\theta) - a(\theta+\omega) \tag{3.24}$$

where b is known and a is not. This can easily be solved by Fourier series; let $\hat{a}(k)$ and $\hat{b}(k)$ be the kth Fourier coefficients of a and b. Then, Eq. (3.24) becomes

$$\sum_{k\in\mathbb{Z}} \hat{b}(k)e^{jk\theta} = \sum_{k\in\mathbb{Z}} \hat{a}(k)e^{jk\theta} - \sum_{k\in\mathbb{Z}} \hat{a}(k)e^{jk(\theta+\omega)} = \sum_{k\in\mathbb{Z}} \hat{a}(k)(1-e^{jk\omega})e^{jk\theta} \tag{3.25}$$

where $j = \sqrt{-1}$. Then, setting $\hat{a}(k) = \hat{b}(k)(1-e^{jk\omega})^{-1}$ allows us to compute $a(\theta)$ except for $\hat{a}(0)$; the formal series for a thus defined will converge on $\mathbb{T}$ for ω sufficiently irrational [47]. Observe that a necessary condition for the existence of a solution is $\hat{b}(0) = 0$; in the $k = 0$ case, $\hat{a}(0)$ cancels out on the right hand side of Eq. (3.24) and can hence take any value, making the solution a non-unique. $\hat{a}(0)$ and $\hat{b}(0)$ are simply the averages of a and b on $\mathbb{T}$.

We first solve Eq. (3.18) for $\hat{\xi}_2(k)$, $k \neq 0$, using the Fourier series method. To set $\hat{\xi}_2(0)$, first find the average α of $\eta_1 - T \times [\xi_2 - \hat{\xi}_2(0)]$. Then, choose $\hat{\xi}_2(0) = \alpha/\hat{T}(0)$; this makes the LHS of Eq. (3.17) have zero average when solving for ξ_1, since

$$\begin{aligned} \int_0^{2\pi} \eta_1 - T\xi_2 \, d\theta &= \int_0^{2\pi} \eta_1 - T\left[\xi_2 - \hat{\xi}_2(0)\right] d\theta - \hat{\xi}_2(0)\int_0^{2\pi} T\, d\theta \\ &= \alpha - \hat{\xi}_2(0)\hat{T}(0) = 0 \end{aligned} \tag{3.26}$$

With ξ_2 fully solved, we then solve Eq. (3.17) for $\hat{\xi}_1(k)$, $k \neq 0$ and arbitrarily choose $\hat{\xi}_1(0) = 0$. Finally, with all four components of ξ solved, we set $K(\theta)$ equal to $K(\theta) + P(\theta)\xi(\theta)$, concluding the K correction part of the quasi-Newton step.

In practice, when solving Eq. (3.18) for ξ_2, we find that the average of $\eta_2(\theta)$, the left hand side of Eq. (3.18), is not exactly zero; we ignore this nonzero average and solve for the $\hat{\xi}_2(k)$ anyways as described earlier. $\hat{\eta}_2(0)$ decreases to zero with each quasi-Newton step as the method converges, so we are able to solve Eq. (3.18) more and more exactly; this is a result of the vanishing lemma of [43], which is applicable since F is an exact symplectic map due to being the fixed-time map of a Hamiltonian system on $\mathbb{R}^4$ [48]. Also, for those familiar with the parameterization method for invariant tori, note that the choice of $\hat{\xi}_1(0) = 0$ takes care of the translation non-uniqueness of solutions of Eq. (3.4) without requiring extra constraint equations.

Remark. *There are methods of numerically solving for ΔK without using P or Λ, including single-shooting [32] and collocation [33]. These methods effectively discretize θ on a grid of N points and solve a linearized version of Eq.* (3.15) *directly for ΔK at those θ values. This requires solving at least a $4N$ dimensional linear system at each correction step. Gaussian elimination applied to this will hence have a computational complexity of $O(N^3)$ and require $O(N^2)$ storage. However, by using P and the nearly diagonal Λ, we decouple the equations and avoid this large dimensional system. The complexity of our quasi-Newton method is $O(N \log N)$ (as some steps use FFT), with $O(N)$ required storage. Furthermore, our method gives not just K, but also the bundle and Floquet matrices P and Λ. The most expensive step in our method is the computation (using numerical integration) of F and DF on the grid of N different values of θ, which is easy to parallelize on the computer.*

3.4.5 Quasi-Newton Step for Correcting P and Λ

Using the newly computed $K(\theta)$, we set the first column of $P(\theta)$ to $DK(\theta)$, and then recompute $DF(K(\theta))$ and $E_{red}(\theta)$ using Eq. (3.13). Finding $\Delta P(\theta)$ and $\Delta\Lambda(\theta)$ to cancel E_{red} then follows similar methodology as ΔK. Let $\Delta P(\theta) = P(\theta)Q(\theta)$; we will solve for Q and $\Delta\Lambda : \mathbb{T} \to \mathbb{R}^{4\times 4}$ satisfying

$$-E_{red}(\theta) = \Lambda(\theta)Q(\theta) - Q(\theta+\omega)\Lambda(\theta) - \Delta\Lambda(\theta) \tag{3.27}$$

Claim. *For ω sufficiently irrational and E_{red} sufficiently small, if Q and $\Delta\Lambda$ solve Eq.* (3.27)*, then adding $\Delta P = PQ$ to P and $\Delta\Lambda$ to Λ reduces E_{red} quadratically.*

Proof. Substitute $P + PQ$ and $\Lambda + \Delta\Lambda$ into Eq. (3.5) to define

$$\begin{aligned}\mathcal{E}(\theta) = DF(K(\theta))[P(\theta) + P(\theta)Q(\theta)] \\ - [P(\theta+\omega) + P(\theta+\omega)Q(\theta+\omega)][\Lambda(\theta) + \Delta\Lambda(\theta)]\end{aligned} \tag{3.28}$$

Using $E_{red}(\theta) = P^{-1}(\theta+\omega)DF(K(\theta))P(\theta) - \Lambda(\theta)$, we then find that

$$\begin{aligned}P(\theta+\omega)^{-1}\mathcal{E}(\theta) &= E_{red}(\theta) + P(\theta+\omega)^{-1}DF(K(\theta))P(\theta)Q(\theta) \\ &\qquad - Q(\theta+\omega)[\Lambda(\theta) + \Delta\Lambda(\theta)] - \Delta\Lambda(\theta) \\ &= E_{red}(\theta) + [\Lambda(\theta) + E_{red}(\theta)]Q(\theta) - Q(\theta+\omega)[\Lambda(\theta) + \Delta\Lambda(\theta)] - \Delta\Lambda(\theta) \\ &= E_{red}(\theta)Q(\theta) - Q(\theta+\omega)\Delta\Lambda(\theta)\end{aligned} \tag{3.29}$$

where the last line follows due to Eq. (3.27). Evaluating Eq. (3.13) with $P + PQ$ and $\Lambda + \Delta\Lambda$ in place of P and Λ and denoting the result as $E_{red,new}$, we have

$$\begin{aligned}E_{red,new}(\theta) &= [P(\theta+\omega) + P(\theta+\omega)Q(\theta+\omega)]^{-1}\mathcal{E}(\theta) \\ &= [I + Q(\theta+\omega)]^{-1}P(\theta+\omega)^{-1}\mathcal{E}(\theta) \\ &= [I + Q(\theta+\omega)]^{-1}[E_{red}(\theta)Q(\theta) - Q(\theta+\omega)\Delta\Lambda(\theta)]\end{aligned} \tag{3.30}$$

Now, for ω sufficiently irrational, Q and $\Delta\Lambda$ will be similar in magnitude to E_{red}. Hence, if E_{red} is small, then $E_{red,new}$ will be quadratically smaller like E_{red}^2. □ □

Since Λ is nearly diagonal, the equations for the different entries of Q and $\Delta\Lambda$ following from Eq. (3.27) are almost completely decoupled from each other. Write

$$E_{red}(\theta) = \begin{bmatrix} E_{LL}(\theta) & E_{LC}(\theta) & E_{LS}(\theta) & E_{LU}(\theta) \\ E_{CL}(\theta) & E_{CC}(\theta) & E_{CS}(\theta) & E_{CU}(\theta) \\ E_{SL}(\theta) & E_{SC}(\theta) & E_{SS}(\theta) & E_{SU}(\theta) \\ E_{UL}(\theta) & E_{UC}(\theta) & E_{US}(\theta) & E_{UU}(\theta) \end{bmatrix}$$

$$Q(\theta) = \begin{bmatrix} 0 & Q_{LC}(\theta) & Q_{LS}(\theta) & Q_{LU}(\theta) \\ 0 & Q_{CC}(\theta) & Q_{CS}(\theta) & Q_{CU}(\theta) \\ 0 & Q_{SC}(\theta) & Q_{SS}(\theta) & Q_{SU}(\theta) \\ 0 & Q_{UC}(\theta) & Q_{US}(\theta) & Q_{UU}(\theta) \end{bmatrix} \qquad \Delta\Lambda(\theta) = \begin{bmatrix} 0 & \Delta T(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta\lambda_s(\theta) & 0 \\ 0 & 0 & 0 & \Delta\lambda_u(\theta) \end{bmatrix} \tag{3.31}$$

As the first column of $P(\theta)$ is fixed to be $DK(\theta)$, we fix the first column of $Q(\theta)$ to be zero so that the first column of ΔP is zero as well. We can then write columns 2-4 of Eq. (3.27) entry by entry to get 12 scalar equations

$$-E_{LC}(\theta) - T(\theta)Q_{CC}(\theta) = Q_{LC}(\theta) - Q_{LC}(\theta+\omega) - \Delta T(\theta) \tag{3.32}$$

$$-E_{LS}(\theta) - T(\theta)Q_{CS}(\theta) = Q_{LS}(\theta) - \lambda_s(\theta)Q_{LS}(\theta+\omega) \tag{3.33}$$

$$-E_{LU}(\theta) - T(\theta)Q_{CU}(\theta) = Q_{LU}(\theta) - \lambda_u(\theta)Q_{LU}(\theta+\omega) \tag{3.34}$$

$$-E_{CC}(\theta) = Q_{CC}(\theta) - Q_{CC}(\theta+\omega) \tag{3.35}$$

$$-E_{CS}(\theta) = Q_{CS}(\theta) - \lambda_s(\theta)Q_{CS}(\theta+\omega) \tag{3.36}$$

$$-E_{CU}(\theta) = Q_{CU}(\theta) - \lambda_u(\theta)Q_{CU}(\theta+\omega) \tag{3.37}$$

$$-E_{SC}(\theta) = \lambda_s(\theta)Q_{SC}(\theta) - Q_{SC}(\theta+\omega) \tag{3.38}$$

$$-E_{SS}(\theta) = \lambda_s(\theta)Q_{SS}(\theta) - \lambda_s(\theta)Q_{SS}(\theta+\omega) - \Delta\lambda_s(\theta) \tag{3.39}$$

$$-E_{SU}(\theta) = \lambda_s(\theta)Q_{SU}(\theta) - \lambda_u(\theta)Q_{SU}(\theta+\omega) \tag{3.40}$$

$$-E_{UC}(\theta) = \lambda_u(\theta)Q_{UC}(\theta) - Q_{UC}(\theta+\omega) \tag{3.41}$$

$$-E_{US}(\theta) = \lambda_u(\theta)Q_{US}(\theta) - \lambda_s(\theta)Q_{US}(\theta+\omega) \tag{3.42}$$

$$-E_{UU}(\theta) = \lambda_u(\theta)Q_{UU}(\theta) - \lambda_u(\theta)Q_{UU}(\theta+\omega) - \Delta\lambda_u(\theta) \tag{3.43}$$

First, we solve Equations (3.36), (3.37), (3.38), (3.40), (3.41), and (3.42), followed by Eq. (3.33) and (3.34) (after back substitution), using the exact same method that was used to solve Eq. (3.19) and (3.20); rearrange each equation so that its solution is the fixed point of an appropriately defined contraction map, which is then iterated to convergence. Such maps always multiply their input by either λ_s or λ_u^{-1}, or both. Eq. (3.35) is solved using the Fourier method of Section 3.4.4; we arbitrarily choose $\hat{Q}_{CC}(0) = 0$. We ignore the nonzero average of E_{CC}, which goes to zero with each quasi-Newton step without affecting method convergence due to F being symplectic (see Appendix A for the proof of this result). Finally, the solutions to Eq. (3.32), (3.39), and (3.43) are non-unique; we choose $Q_{LC} = Q_{SS} = Q_{UU} = 0$, so that we have $\Delta T(\theta) = E_{LC}(\theta) + T(\theta)Q_{CC}(\theta)$, $\Delta\lambda_s(\theta) = E_{SS}(\theta)$, and $\Delta\lambda_u(\theta) = E_{UU}(\theta)$.

Once Q and $\Delta\Lambda$ are known, we set $P(\theta)$ equal to $P(\theta) + P(\theta)Q(\theta)$, $\Lambda(\theta)$ equal to $\Lambda(\theta) + \Delta\Lambda(\theta)$, and then recompute $E(\theta)$ and $E_{red}(\theta)$ using Eq. (3.12) and (3.13). Finally, we go back to the quasi-Newton step for correcting the torus parameterization $K(\theta)$ and repeat the entire method until E and E_{red} are within tolerance.

Remark. *If T, λ_s, and λ_u are constant, we can choose the non-unique solutions of Eq.* (3.32)*,* (3.39)*, and* (3.43) *such that they remain constant. In particular, choose ΔT, $\Delta\lambda_s$, and $\Delta\lambda_u$ as the (constant) averages of E_{LC}, E_{SS} and E_{UU}, respectively, and solve for Q_{LC}, Q_{SS}, and Q_{UU} using Fourier methods. Our experience was that this choice of solution negatively affected the numerical stability of our method, however; thus, we did not keep T, λ_s, and λ_u constant in our implementation.*

3.4.6 A Remark on Convergence

The focus of this chapter is to specify the algorithms, provide details of implementation, and give practical results of the implementation in physical problems. Nevertheless, we wish to mention that there are results which rigorously prove that our algorithm converges when given initial K, P, Λ with small enough error (depending on some condition numbers). Due to the practical focus of this chapter, we will not go into detail, but we want to give a flavor of the argument. For readers interested primarily in applications, this section can be skipped.

The convergence is due to the so-called Kolmogorov-Arnold-Moser (KAM) theory, which is a very far reaching generalization of the Newton method. In particular, we take advantage of the recent developments in a-posteriori versions of KAM theory [43], which does not require an integrable system, only approximate solutions of functional equations. We present some salient features. For any analytic function of an angle $u : \mathbb{T} \to \mathbb{C}^n$, define $||u||_\rho = \sup_{|\operatorname{Im} z| \leq \rho} |u(z)|$. It is possible to show [47] that the solutions of (3.24) satisfy $||a||_{\rho-\delta} \leq C_1 \delta^{-\tau} ||b||_\rho$ for some $C_1, \tau > 0$. That is, if the right hand side is analytic in a certain complex domain, the solution is analytic in a slightly smaller complex domain, and we have estimates of the size in terms of the domain lost; note that both domains contain all real angles from 0 to 2π, which is what we are actually interested in. The well known Cauchy estimates [49] for derivatives of a function in a slightly smaller domain have the same form.

The formal procedure we have given indeed reduces E and E_{red}, to something quadratically smaller, but, performing the estimates with care, only in a slightly smaller complex domain. Denoting the invariance error and the reducibility errors after one quasi-Newton step by E_{new} and $E_{red,new}$, we have that

$$\|E_{new}\|_{\rho-\delta} + \|E_{red,new}\|_{\rho-\delta} \leq C_2 \delta^{-2\tau-2} \left(\|E\|_\rho + \|E_{red}\|_\rho\right)^2 \tag{3.44}$$

for some $C_2 > 0$. There are standard arguments in KAM theory (*"hard implicit function theorems"*, see [45]) which show that, given an algorithm satisfying Eq. (3.44) and a sufficiently small initial error, the algorithm step can be iterated infinitely many times to convergence in a domain slightly smaller than the original. These estimates also show that the final answer is close to the initial approximation of K, P, Λ if the initial error is small enough.

3.4.7 Modifying P for Constant Λ

Let K, P, and Λ be a solution to Eq. (3.4)-(3.5). For purposes of numerical stability as well as stable/unstable manifold computation (see Section 3.5), it can be useful to modify columns 2, 3, and 4 of P in such a way that $\Lambda = P^{-1}(\theta+\omega)DF(K(\theta))P(\theta)$ becomes a constant matrix of the form in Eq. (3.6), i.e. $T(\theta)$, $\lambda_s(\theta)$, and $\lambda_u(\theta)$ become constant. This can also enable the usage of Fourier methods instead of fixed point iteration during the quasi-Newton method (see Remarks 3.4.4 and 3.4.5).

For columns 3 and 4 of P, the stable and unstable bundles $\mathbf{v}_s(\theta)$ and $\mathbf{v}_u(\theta)$ respectively, just a simple rescaling is needed to make $\lambda_s(\theta)$ and $\lambda_u(\theta)$ constant. Let $\bar{\lambda}_s, \bar{\lambda}_u \in \mathbb{R}$ and

$a_s, a_u : \mathbb{T} \to \mathbb{R}$ be the solutions to

$$\log(\lambda_s(\theta)) - \log(\bar{\lambda}_s) = \log(a_s(\theta + \omega)) - \log(a_s(\theta)) \tag{3.45}$$

$$\log(\lambda_u(\theta)) - \log(\bar{\lambda}_u) = \log(a_u(\theta + \omega)) - \log(a_u(\theta)) \tag{3.46}$$

We choose $\bar{\lambda}_s = \exp\left[\frac{1}{2\pi}\int_0^{2\pi} \log(\lambda_s(\theta))\, d\theta\right]$ so the LHS of Eq. (3.45) has zero average. Letting $u(\theta) = \log(a_s(\theta))$, Eq. (3.45) becomes a cohomological equation of form Eq. (3.24) which can be solved for u by the Fourier series method; we choose $\hat{u}(0) = 0$. This gives $a_s(\theta) = e^{u(\theta)}$. We can solve Eq. (3.46) for $a_u(\theta)$ in the exact same manner. Finally, one can replace columns 3 and 4 of P by $a_s(\theta)\mathbf{v}_s(\theta)$ and $a_u(\theta)\mathbf{v}_u(\theta)$, and replace $\lambda_s(\theta)$ and $\lambda_u(\theta)$ in Λ by $\bar{\lambda}_s$ and $\bar{\lambda}_u$. We prove that this works now.

Lemma 2. *If $\mathbf{v}_s(\theta)$ and $\mathbf{v}_u(\theta)$ satisfy Eq.* (3.9)-(3.10)*, and $a_s(\theta)$ and $a_u(\theta)$ satisfy Eq.* (3.45)-(3.46)*, then $\mathbf{v}_{s,new}(\theta) = a_s(\theta)\mathbf{v}_s(\theta)$ and $\mathbf{v}_{u,new}(\theta) = a_u(\theta)\mathbf{v}_u(\theta)$ satisfy*

$$DF(K(\theta))\mathbf{v}_{s,new}(\theta) = \bar{\lambda}_s \mathbf{v}_{s,new}(\theta + \omega) \tag{3.47}$$

$$DF(K(\theta))\mathbf{v}_{u,new}(\theta) = \bar{\lambda}_u \mathbf{v}_{u,new}(\theta + \omega) \tag{3.48}$$

Proof. We prove the result for $\mathbf{v}_{s,new}$; the case of $\mathbf{v}_{u,new}$ can be proven in the exact same manner. Since $\mathbf{v}_s(\theta)$ satisfies Eq. (3.9), we have

$$\begin{aligned} DF(K(\theta))\mathbf{v}_{s,new}(\theta) &= DF(K(\theta))a_s(\theta)\mathbf{v}_s(\theta) = a_s(\theta)\lambda_s(\theta)\mathbf{v}_s(\theta + \omega) \\ &= a_s(\theta + \omega)\bar{\lambda}_s \mathbf{v}_s(\theta + \omega) = \bar{\lambda}_s \mathbf{v}_{s,new}(\theta + \omega) \end{aligned} \tag{3.49}$$

where $a_s(\theta)\lambda_s(\theta) = a_s(\theta + \omega)\bar{\lambda}_s$ follows from exponentiating Eq. (3.45). □ □

We can also modify the second column of P, the symplectic conjugate center direction $\mathbf{v}_c(\theta)$, to make $T(\theta)$ constant. This is possible because as mentioned in Section 3.4.2, the symplectic conjugate is not unique; given $a : \mathbb{T} \to \mathbb{R}$ and $\mathbf{v}_c(\theta)$ satisfying Eq. (3.8), the function $\mathbf{v}_c(\theta) + a(\theta)DK(\theta)$ also solves Eq. (3.8) except with a change in $T(\theta)$ (which was anyways arbitrary). Hence, we choose $a(\theta)$ which kills all variation of $T(\theta)$ about its average $\hat{T}(0)$. The equation for this is:

$$-(T(\theta) - \hat{T}(0)) = a(\theta) - a(\theta + \omega) \tag{3.50}$$

which can be solved using the Fourier series method given for Eq. (3.24). Then, one simply adds $a(\theta)DK(\theta)$ to column 2 of P and replaces $T(\theta)$ with $\hat{T}(0)$ in Λ. Note that the LHS of Eq. (3.50) has average zero, so a solution $a(\theta)$ can be found.

Lemma 3. *If $\mathbf{v}_c(\theta)$ and $a(\theta)$ satisfy Eq.* (3.8) *and* (3.50)*, respectively, then the function $\mathbf{v}_{c,new}(\theta) = \mathbf{v}_c(\theta) + a(\theta)DK(\theta)$ is also a symplectic conjugate and satisfies*

$$DF(K(\theta))\mathbf{v}_{c,new}(\theta) = \hat{T}(0)DK(\theta + \omega) + \mathbf{v}_{c,new}(\theta + \omega) \tag{3.51}$$

Proof. Since $\mathbf{v}_c(\theta)$ satisfies Eq. (3.8) and $DK(\theta)$ satisfies Eq. (3.11), we have

$$\begin{aligned} DF(K(\theta))\mathbf{v}_{c,new}(\theta) &= DF(K(\theta))\left[\mathbf{v}_c(\theta) + a(\theta)DK(\theta)\right] \\ &= \left[T(\theta) + a(\theta)\right] DK(\theta + \omega) + \mathbf{v}_c(\theta + \omega) \\ &= \left[\hat{T}(0) + a(\theta + \omega)\right] DK(\theta + \omega) + \mathbf{v}_c(\theta + \omega) \\ &= \hat{T}(0)DK(\theta + \omega) + \mathbf{v}_{c,new}(\theta + \omega) \end{aligned} \tag{3.52}$$

where the relation $T(\theta) + a(\theta) = \hat{T}(0) + a(\theta + \omega)$ follows from Eq. (3.50). □ □

3.4.8 Initialization for Continuation by ε

To compute invariant circles and bundles of stroboscopic maps in periodically-perturbed PCRTBP models with some desired perturbation parameter $\varepsilon_f > 0$, we start from periodic orbits and their bundles in the unperturbed PCRTBP ($\varepsilon = 0$) and continue by ε until the torus and bundles for $\varepsilon = \varepsilon_f$ are found. Our quasi-Newton method-based continuation follows the standard procedure; choose a number of continuation steps n, take an invariant circle and bundles from the $\varepsilon = 0$ system, and use them as an initial guess for the quasi-Newton method to solve for the circle and bundles in the $\varepsilon = \varepsilon_f/n$ system. Similarly, for $i = 0, \dots, n-1$, use the solution from the $\varepsilon_f i/n$ system as an initial guess for the solution in the $\varepsilon_f(i+1)/n$ system. Once $i = n-1$, we have the torus and bundles for $\varepsilon = \varepsilon_f$. We need to find the $\varepsilon = 0$ solution to initialize the continuation, however.

To get $K(\theta)$, $P(\theta)$, and $\Lambda(\theta)$ solving Eq. (3.4) and (3.5) for the $\varepsilon = 0$ PCRTBP case, one needs to first choose a periodic orbit which is to be continued (recall that PCRTBP periodic orbits are also invariant circles of the stroboscopic map $F = F_{\varepsilon=0}$, unless the orbit period is resonant with $2\pi/\Omega_p$). From this periodic orbit, we get its period T_1 and hence the rotation number $\omega = 4\pi^2/(T_1\Omega_p)$, as well as a point $\mathbf{x}_0$ lying on the orbit. Let $\phi(\mathbf{x}, t)$ denote the time-t map of the point $\mathbf{x} \in \mathbb{R}^4$ by the PCRTBP equations of motion. Then, we can take $K(\theta) = \phi(\mathbf{x}_0, T_1\frac{\theta}{2\pi})$ if the periodic orbit monodromy matrix stable and unstable eigenvalues are positive. If they are negative, though, the "double covering" trick of [50] needs be used so that $P(\theta)$ can be continuously defined (as the stable/unstable bundles are Möbius strips in this case). For this, set $K(\theta) = \phi(\mathbf{x}_0, 2T_1\frac{\theta}{2\pi})$ and $\omega = 2\pi^2/(T_1\Omega_p)$ so that K sweeps over the periodic orbit twice as θ goes from 0 to 2π. In either case, it is easy to verify that $K(\theta)$ satisfies Eq. (3.4).

Next, set the first column of $P(\theta)$ to be $DK(\theta)$, and set the third and fourth columns of $P(\theta)$ as the stable and unstable unit eigenvectors of the periodic orbit monodromy matrix at the point $K(\theta)$. Denote these stable and unstable eigenvectors as $\mathbf{v}_s(\theta)$ and $\mathbf{v}_u(\theta)$, respectively. One needs to make sure that the directions of $\mathbf{v}_s(\theta)$ and $\mathbf{v}_u(\theta)$ at each point $K(\theta)$ are chosen such that they are continuously oriented functions of θ; this is always possible if K is defined as previously described. Finally, finding the second column of P requires some extra calculations.

As mentioned in Section 3.4.2, the second column of P represents the symplectic conjugate direction to $DK(\theta)$ and is part of the center bundle. The first step in its computation is to compute $\lambda_s(\theta)$ and $\lambda_u(\theta) : \mathbb{T} \to \mathbb{R}$ such that

$$DF(K(\theta))\mathbf{v}_s(\theta) = \lambda_s(\theta)\mathbf{v}_s(\theta + \omega) \tag{3.53}$$

$$DF(K(\theta))\mathbf{v}_u(\theta) = \lambda_u(\theta)\mathbf{v}_u(\theta + \omega) \tag{3.54}$$

which can be done since $DF(K(\theta))$ maps the stable and unstable bundles into themselves. Next, find functions $A(\theta), B(\theta), C(\theta)$, and $D(\theta) : \mathbb{T} \to \mathbb{R}$ such that

$$\begin{aligned} DF(K(\theta))\frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2} = A(\theta)DK(\theta+\omega) + B(\theta)\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2} \\ + C(\theta)\mathbf{v}_s(\theta+\omega) + D(\theta)\mathbf{v}_u(\theta+\omega) \end{aligned} \tag{3.55}$$

where $J = \begin{bmatrix} 0_{2\times2} & I_{2\times2} \\ -I_{2\times2} & 0_{2\times2} \end{bmatrix}$ is the 4×4 matrix of the symplectic form in the usual Euclidean metric on $\mathbb{R}^4$. All the quantities in (3.55) are known except for A, B, C, and D. We can therefore consider Eq. (3.55) as a system of linear equations for A, B, C, and D which can be solved. One will find that $B(\theta) = 1$; this occurs as a result of symplectic geometric considerations (see Eq. (3.64)). After this, we solve for functions $f_1(\theta), f_2(\theta) : \mathbb{T} \to \mathbb{R}$ such that

$$C(\theta) = f_1(\theta+\omega) - \lambda_s(\theta)f_1(\theta) \tag{3.56}$$

$$D(\theta) = f_2(\theta+\omega) - \lambda_u(\theta)f_2(\theta) \tag{3.57}$$

which can be done using the same contraction map iteration method used to solve Equations (3.19) and (3.20) in Section 3.4.4. Finally, we can express the second column of $P(\theta)$, the symplectic conjugate direction $\mathbf{v}_c(\theta)$, as

$$\mathbf{v}_c(\theta) = \frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2} + f_1(\theta)\mathbf{v}_s(\theta) + f_2(\theta)\mathbf{v}_u(\theta) \tag{3.58}$$

With $P(\theta)$ known, to find Λ one can simply use $\Lambda(\theta) = P^{-1}(\theta+\omega)DF(K(\theta))P(\theta)$, after which we can start the continuation. As long as the previous steps were followed correctly, Λ will be of the form given in Eq. (3.6). To see this, recall the discussion in Section 3.4.2, and note that the first, third, and fourth columns of P satisfy Equations (3.7), (3.9), and (3.10). Hence, we just need to show that the second column of P satisfies Eq. (3.8). We prove this now.

Lemma 4. *For some $T : \mathbb{T} \to \mathbb{R}$. the function $\mathbf{v}_c(\theta)$ defined in Eq.* (3.58) *satisfies*

$$DF(K(\theta))\mathbf{v}_c(\theta) = T(\theta)DK(\theta + \omega) + \mathbf{v}_c(\theta + \omega) \tag{3.59}$$

Proof. Applying Eq. (3.58) and then Equations (3.53), (3.54), and (3.55), we have

$$\begin{aligned} DF(K(\theta))\mathbf{v}_c(\theta) = DF(K(\theta)) &\left(\frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2} + f_1(\theta)\mathbf{v}_s(\theta) + f_2(\theta)\mathbf{v}_u(\theta) \right) \\ =& A(\theta)DK(\theta+\omega) + B(\theta)\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2} \\ &+ \left(C(\theta) + \lambda_s(\theta)f_1(\theta)\right)\mathbf{v}_s(\theta+\omega) + \left(D(\theta) + \lambda_u(\theta)f_2(\theta)\right)\mathbf{v}_u(\theta+\omega) \end{aligned} \tag{3.60}$$

Recalling Equations (3.56) and (3.57), we thus have that

$$\begin{aligned} DF(K(\theta))\mathbf{v}_c(\theta) =& A(\theta)DK(\theta+\omega) + B(\theta)\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2} \\ &+ f_1(\theta+\omega)\mathbf{v}_s(\theta+\omega) + f_2(\theta+\omega)\mathbf{v}_u(\theta+\omega) \end{aligned} \tag{3.61}$$

Flow maps of Hamiltonian systems are symplectic [51]. Hence, F satisfies $\Omega(\mathbf{v}_1, \mathbf{v}_2) = \Omega(DF(K(\theta))\mathbf{v}_1, DF(K(\theta))\mathbf{v}_2)$ for all $\mathbf{v}_1$, $\mathbf{v}_2 \in \mathbb{R}^4$, where Ω is the bilinear symplectic form defined on Euclidean $\mathbb{R}^4$ as $\Omega(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T J \mathbf{v}_2$. It is easy to see that $\Omega(\mathbf{v}_1, \mathbf{v}_1) = 0$ for any $\mathbf{v}_1 \in \mathbb{R}^4$. Furthermore, defining $L = \max_{\theta \in \mathbb{T}} |\lambda_s(\theta)| < 1$ and recalling equations

(3.11) and (3.53), we have that

$$\begin{aligned}\max_{\theta\in\mathbb{T}}|\Omega(DK(\theta),\mathbf{v}_s(\theta))| &= \max_{\theta\in\mathbb{T}}|\Omega\left(DF(K(\theta))DK(\theta),DF(K(\theta))\mathbf{v}_s(\theta)\right)|\\ &= \max_{\theta\in\mathbb{T}}|\Omega\left(DK(\theta+\omega),\lambda_s(\theta)\mathbf{v}_s(\theta+\omega)\right)|\\ &= \max_{\theta\in\mathbb{T}}|\lambda_s(\theta)|\,|\Omega\left(DK(\theta+\omega),\mathbf{v}_s(\theta+\omega)\right)|\\ &\le L\max_{\theta\in\mathbb{T}}|\Omega\left(DK(\theta+\omega),\mathbf{v}_s(\theta+\omega)\right)| = L\max_{\theta\in\mathbb{T}}|\Omega\left(DK(\theta),\mathbf{v}_s(\theta)\right)|\end{aligned} \tag{3.62}$$

which implies that $\max_{\theta\in\mathbb{T}}|\Omega\left(DK(\theta),\mathbf{v}_s(\theta)\right)| = 0$ since $0 < L < 1$. Thus, for all $\theta\in\mathbb{T}$, $\Omega\left(DK(\theta),\mathbf{v}_s(\theta)\right) = 0$. We can also show that $\Omega\left(DK(\theta),\mathbf{v}_u(\theta)\right) = 0$ in a very similar manner to Eq. (3.62). Hence, using Eq. (3.58) for $\mathbf{v}_c$, we find

$$\begin{aligned}\Omega(DK(\theta),\mathbf{v}_c(\theta)) &= \Omega\left(DK(\theta),\frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2}+f_1(\theta)\mathbf{v}_s(\theta)+f_2(\theta)\mathbf{v}_u(\theta)\right)\\ &= \Omega\left(DK(\theta),\frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2}\right)\\ &= DK(\theta)^TJ\frac{J^{-1}DK(\theta)}{\|DK(\theta)\|^2} = \frac{DK(\theta)^TDK(\theta)}{\|DK(\theta)\|^2} = 1\end{aligned} \tag{3.63}$$

Since F is a symplectic map, using Eq. (3.61) we have that

$$\begin{aligned}1 &= \Omega(DK(\theta),\mathbf{v}_c(\theta))\\ &= \Omega\left(DF(K(\theta))DK(\theta),DF(K(\theta))\mathbf{v}_c(\theta)\right)\\ &= \Omega\Bigg(DK(\theta+\omega),A(\theta)DK(\theta+\omega)+B(\theta)\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2}\\ &\qquad +f_1(\theta+\omega)\mathbf{v}_s(\theta+\omega)+f_2(\theta+\omega)\mathbf{v}_u(\theta+\omega)\Bigg)\\ &= \Omega\left(DK(\theta+\omega),B(\theta)\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2}\right)\\ &= B(\theta)DK(\theta+\omega)^TJ\frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2} = B(\theta)\end{aligned} \tag{3.64}$$

proving that $B(\theta) = 1$. Therefore, substituting this into Eq. (3.61) gives

$$\begin{aligned} DF(K(\theta))\mathbf{v}_c(\theta) =& A(\theta)DK(\theta+\omega) + \frac{J^{-1}DK(\theta+\omega)}{\|DK(\theta+\omega)\|^2} \\ &+ f_1(\theta+\omega)\mathbf{v}_s(\theta+\omega) + f_2(\theta+\omega)\mathbf{v}_u(\theta+\omega) \end{aligned} \tag{3.65}$$

Finally, we see from Eq. (3.58) that the last 3 terms on the RHS of Eq. (3.65) are precisely $\mathbf{v}_c(\theta+\omega)$. Letting $T(\theta) = A(\theta)$, we hence conclude that

$$DF(K(\theta))\mathbf{v}_c(\theta) = T(\theta)DK(\theta+\omega) + \mathbf{v}_c(\theta+\omega) \tag{3.66}$$

which is what we sought to prove. □ □

3.4.9 Computing Families of Tori: Continuation by ω

The continuation by ε described in Section 3.4.8 is carried out with ω fixed. However, in the PCRTBP, periodic orbits occur in one-parameter families, with varying rotation numbers ω under $F_{\varepsilon=0}$. The same is true of invariant circles of F_ε for $\varepsilon = \varepsilon_f > 0$ as well. To compute the family of F_{ε_f}-invariant tori corresponding to a PCRTBP periodic orbit family, one option is to continue several different periodic orbits from that family (corresponding to different ω values) by ε. However, this is inefficient, as the tori and bundles computed for $\varepsilon < \varepsilon_f$ are not of interest. Instead, it is better to first compute just one invariant circle of F_{ε_f}, along with its bundles, at some rotation number $\omega = \omega_0$ using continuation by ε. After this, one can continue the ω_0 circle/bundles by ω, with $\varepsilon = \varepsilon_f$ fixed. The continuation by ω is quite similar to the continuation by ε; given a known exact solution to Eq. (3.4) and (3.5) for $\omega = \omega_i$, $i \in \mathbb{Z}$, one uses this to form an initial guess for the quasi-Newton method to compute the torus/bundles for $\omega = \omega_{i+1} = \omega_i + \Delta\omega_i$. This recursively gives us tori for a range of ω values. The $\Delta\omega_i$ are called the continuation step sizes.

One can use the torus and bundles for $\omega = \omega_i$ directly as an initial guess for $\omega = \omega_i + \Delta\omega_i$. However, it is extremely easy to use K, P, and Λ from ω_i to compute a better initial guess for the $\omega_i + \Delta\omega_i$ torus, which aids in quasi-Newton method convergence. Assume that Λ has constant $T(\theta) = T$ (apply the procedure from Section 3.4.7 to P and Λ if necessary). Then, using $\mathbf{v}_c(\theta)$ to denote column 2 of P, the initial guess for the $\omega_i + \Delta\omega_i$ torus parameterization should be $K_{new}(\theta) = K(\theta) + (\Delta\omega_i/T)\mathbf{v}_c(\theta)$. We justify this now.

Claim. *If K, P, and Λ (with Λ constant) solve Eq.* (3.4)-(3.5) *for $\omega = \omega_i$, then $K_{new}(\theta) = K(\theta) + (\Delta\omega_i/T)\mathbf{v}_c(\theta)$ solves Eq.* (3.4) *for $\omega = \omega_i + \Delta\omega_i$ up to $\mathcal{O}(\Delta\omega_i^2)$.*

Proof. For notational convenience, write ω and $\Delta\omega$ in place of ω_i and $\Delta\omega_i$, respectively. Then, evaluating $F(K_{new}(\theta)) - K_{new}(\theta + \omega + \Delta\omega)$, we find this equals

$$\begin{aligned} &F\bigg(K(\theta) + \frac{\Delta\omega}{T}\mathbf{v}_c(\theta)\bigg) - \bigg[K(\theta+\omega+\Delta\omega) + \frac{\Delta\omega}{T}\mathbf{v}_c(\theta+\omega+\Delta\omega)\bigg] \\ &\quad = F(K(\theta)) + DF(K(\theta))\frac{\Delta\omega}{T}\mathbf{v}_c(\theta) + \mathcal{O}(\Delta\omega^2) \\ &\qquad\qquad - \bigg[K(\theta+\omega) + \Delta\omega DK(\theta+\omega) + \frac{\Delta\omega}{T}\mathbf{v}_c(\theta+\omega) + \mathcal{O}(\Delta\omega^2)\bigg] \\ &\quad = \frac{\Delta\omega}{T}\left[DF(K(\theta))\mathbf{v}_c(\theta) - T\,DK(\theta+\omega) - \mathbf{v}_c(\theta+\omega)\right] + \mathcal{O}(\Delta\omega^2) = \mathcal{O}(\Delta\omega^2) \end{aligned} \tag{3.67}$$

where the last equality follows from Eq. (3.8). □ □

Remark. *Using the Poincaré-Lindstedt method, it is possible to get higher order expansions in $\Delta\omega_i$ for K_{new} than the linear approximation $K(\theta) + (\Delta\omega_i/T)\mathbf{v}_c(\theta)$. This requires significant extra computations which we decided not to carry out.*

There is one more difference between continuation by ω and continuation by ε. The continuation by ε uses a fixed step size ε_f/n. However, for continuation by ω, the step size $\Delta\omega_i$ must be varied due to quasi-Newton method divergence for insufficiently irrational ω. At such ω values, the PCRTBP invariant circle breaks down after the perturbation $\varepsilon = \varepsilon_f$, leading to a gap between tori at smaller and larger rotation numbers. Our continuation needs to "jump" over this gap. Suppose we have a torus and bundles for $\omega = \omega_i$, and let φ_i denote the largest of $\Delta\omega_{i-1}, \Delta\omega_{i-2}, \ldots, \Delta\omega_{i-5}$. It is natural to try $\Delta\omega_i = \varphi_i$. If the quasi-Newton method diverges for $\omega = \omega_i + \varphi_i$, however, then instead one can try $\Delta\omega_i = \varphi_i/2$;

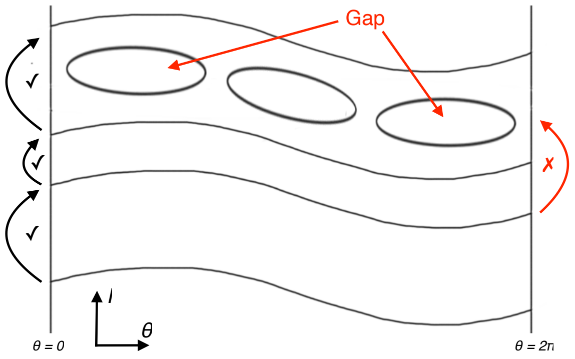


Figure 3.1: Schematic of crossing gaps during ω continuation (consider the $\theta = 0$ and $\theta = 2\pi$ lines to be glued together to form a cylinder)

if this still does not work, try $\Delta\omega_i = \varphi_i/2^2$, and so on until we find a $\Delta\omega_i$ that works. Once we have the circle/bundles for $\omega_{i+1} = \omega_i + \Delta\omega_i$, we repeat the process.

In this procedure, it is possible for $\Delta\omega_{i+1}$ to be larger than $\Delta\omega_i$, which is what allows us to "jump" over gaps in the tori. For example, suppose that $\Delta\omega_{i-1}$ through $\Delta\omega_{i-5}$ are all equal to $\phi = \frac{1+\sqrt{5}}{2} \times 10^{-4}$, so $\varphi_i = \phi$. Then, it can happen that the quasi-Newton method diverges for $\omega = \omega_i + \phi$, but converges for $\omega = \omega_i + \phi/2 = \omega_{i+1}$, so $\Delta\omega_i = \phi/2$. However, φ_{i+1} will still equal ϕ, so we try $\Delta\omega_{i+1} = \phi$. If the quasi-Newton method converges for $\omega = \omega_{i+1} + \phi = \omega_i + 3\phi/2$, then we will have crossed the torus gap encountered earlier at $\omega = \omega_i + \phi$. This is schematically illustrated in Fig. 3.1; we draw the tori on a projection of the 2D cylindrical NHIM Ξ_ε defined in Section 3.3.1 (we let (θ, I) be coordinates on Ξ_ε).

3.4.10 Discretization and Implementation

When implementing the previously-described methods on a computer, it is necessary to discretize all the functions used as well as the operations on them. We represent K, P, Λ, and other functions of θ as arrays of their values on a discrete grid of N evenly spaced θ values $\theta_i = 2\pi i/N$, $i = 0, \ldots, N-1$. Many operations on functions can be carried out element-wise on these arrays; such operations include basic scalar arithmetic, matrix multiplication, and matrix inversion. For instance, given arrays of values $P(\theta_i)$ and $\xi(\theta_i)$, we can calculate a new array of N values $\Delta K(\theta_i) = P(\theta_i)\xi(\theta_i)$ (note that the ΔK array will actually contain $4N$ floating point numbers, since each $\Delta K(\theta_i) \in \mathbb{R}^4$).

Other operations are more efficiently carried out using Fourier coefficients. For instance, given an array of function values $a(\theta_i)$ for some $a : \mathbb{T} \to \mathbb{R}$, we can use

$$a(\theta_i) = \frac{1}{N}\sum_{k=0}^{N-1} \hat{a}(k)e^{jk\theta_i} \to a(\theta_i + \omega) = \frac{1}{N}\sum_{k=0}^{N-1}[\hat{a}(k)e^{jk\omega}]e^{jk\theta_i} \tag{3.68}$$

$$a(\theta_i) = \frac{1}{N}\sum_{k=0}^{N-1} \hat{a}(k)e^{jk\theta_i} \to Da(\theta_i) = \frac{1}{N}\sum_{k=0}^{N-1}[jk\hat{a}(k)]e^{jk\theta_i} \tag{3.69}$$

to translate or differentiate a. We use the fast Fourier transform (FFT) to get N Fourier coefficients $\hat{a}(k)$, multiply each $\hat{a}(k)$ by $e^{jk\omega}$ (translation) or jk (differentiation), and then take the inverse FFT to get an array of values $a(\theta_i + \omega)$ or $Da(\theta_i)$. Solving cohomological equations like (3.24) also requires working with Fourier coefficients as described in Section 3.4.4.

A few numerical problems were experienced due to the discretization of continuous functions on the computer. Let $\text{Trans}_\omega(a(\theta_i))$ represent the array of $a(\theta_i + \omega)$ values found by applying the algorithm of Eq. (3.68) to the array of $a(\theta_i)$ values. The first problem was that $\text{Trans}_\omega(F(\theta_i)G(\theta_i)) \neq \text{Trans}_\omega(F(\theta_i)) \times \text{Trans}_\omega(G(\theta_i))$; multiplying two arrays and then translating the result gives a different result than first translating the two arrays and then multiplying the results. Also, $F(\theta_i) \neq \text{Trans}_{-\omega}(\text{Trans}_{+\omega}(F(\theta_i)))$ when N is even and real-to-complex FFT is used (for real data). These issues can prevent quasi-Newton method convergence.

Both inequalities are most pronounced when the high-frequency discrete Fourier transform coefficients of F or G are large in magnitude. It is not expected that the tori or bundles we compute should have significant high-frequency oscillations as a function of θ. Hence, a solution to these two problems was to run $K(\theta_i)$ and $P(\theta_i)$ through a lowpass filter during the first two or three quasi-Newton steps, as well as when the quasi-Newton method would start diverging; note that this is somewhat reminiscent of Arnold's use of truncated Fourier series with successively increasing cutoff frequencies in his proof of the KAM theory [45]. We also found that modifying P between continuation steps to make Λ constant, as described in Section 3.4.7, greatly mitigates these numerical issues as well. Finally, if the high-frequency Fourier coefficients keep becoming large after each quasi-Newton step despite filtering and constant Λ, we increase the number of Fourier modes used (equivalent to increasing N).

One phenomenon we noticed during the implementation of our quasi-Newton method-based continuation was that generally, the torus parameterization $K(\theta)$ converges to within a given error tolerance before the bundle and Floquet matrices $P(\theta)$ and $\Lambda(\theta)$. We can use this to further improve the numerical stability of our quasi-Newton method, by using the converged $K(\theta)$ to directly compute the full P and Λ matrices. To do this, as mentioned earlier, the first column of P is simply $DK(\theta)$. The third and fourth columns of P (the stable and unstable bundles $\mathbf{v}_s$ and $\mathbf{v}_u$) can be found using the "power method", as is described by [52]. For this, first set $\mathbf{v}_{s,0}(\theta)$ and $\mathbf{v}_{u,0}(\theta)$ equal to the unit-length normalized third and fourth columns of $P(\theta)$ (the unconverged, approximate stable and unstable bundles). This is then followed by the iteration

$$\mathbf{v}_{s,i+1}(\theta) = \frac{DF(K(\theta))^{-1}\mathbf{v}_{s,i}(\theta+\omega)}{\|DF(K(\theta))^{-1}\mathbf{v}_{s,i}(\theta+\omega)\|} \tag{3.70}$$

$$\mathbf{v}_{u,i+1}(\theta) = \frac{DF(K(\theta-\omega))\mathbf{v}_{u,i}(\theta-\omega)}{\|DF(K(\theta-\omega))\mathbf{v}_{u,i}(\theta-\omega)\|} \tag{3.71}$$

which should converge after a few iterations (in practice, we also run each $\mathbf{v}_{s,i}(\theta)$, $\mathbf{v}_{u,i}(\theta)$ through a lowpass filter after its computation). Once the iterations have converged, we use the methods of Section 3.4.7 to rescale $\mathbf{v}_s$ and $\mathbf{v}_u$ to ensure constant λ_s and λ_u. Finally, we can use the exact same method presented in Section 3.4.8 to compute the second column of P from the known DK, $\mathbf{v}_s$, and $\mathbf{v}_u$ (see Eq. (3.55)-(3.58)); the method of Section 3.4.7 is then applied to make T constant. This gives us the final P and Λ matrices which in our experience not only usually satisfy Eq. (3.5) to within tolerance (sometimes one last quasi-Newton correction step is required), but also have smaller high-order Fourier coefficients than the earlier approximate P and Λ; this further improves our method's numerical stability.

3.4.11 Numerical Results in the PERTBP

We implemented and successfully applied the methods described in the previous sections to the computation of invariant circles and their bundles for the Jupiter-Europa PERTBP stroboscopic map. We used a tolerance of 10^{-7} in Eq. (3.4)-(3.5). The circles and bundles were found by first continuing Jupiter-Europa PCRTBP unstable resonant periodic orbits by eccentricity ε to $\varepsilon = 0.0094$ (the real value) for fixed ω, and then continuing the circles and bundles by ω while fixing $\varepsilon = 0.0094$.

Both 3:4 as well as 5:6 resonant tori were computed. Fig. 3.2 shows the continuation by eccentricity of a 5:6 resonant periodic orbit from the PCRTBP to an invariant circle of the PERTBP stroboscopic map; for this, we used $N = 2048$ discretization θ_i values. The plot on the right zooms into the region near Europa; the leftmost curve there corresponds to $\varepsilon = 0.0094$, which is to be expected as Europa's periapsis moves leftwards as ε increases. Fig. 3.3 shows the continuation of a 3:4 resonant torus in the physical $\varepsilon = 0.0094$ Jupiter-Europa PERTBP by ω, which yields a family of resonant tori in the system for $\omega \in [1.536217, 1.567314]$; N ranged from 1024 to 32768, with larger N required for tori passing close to the singularity in the equations of motion at Europa. Fig. 3.4 shows a family of Jupiter-Europa PERTBP 5:6 resonant tori, also generated using continuation by ω. This family, like the PCRTBP 5:6 resonant orbit family, does not have monotonically increasing or decreasing rotation numbers; ω starts at 1.035166 for the leftmost torus in the zoomed-in plot, decreases to 1.027137, and then increases to 1.040911 for the rightmost torus. Thus, we first continued two different PCRTBP 5:6 resonant orbits by ε to get two PERTBP tori, one in each of the two sections of tori with monotone ω. These two tori were then continued by ω to sweep out the tori in their corresponding sections. For this case, N ranged from 1024 to 4096.

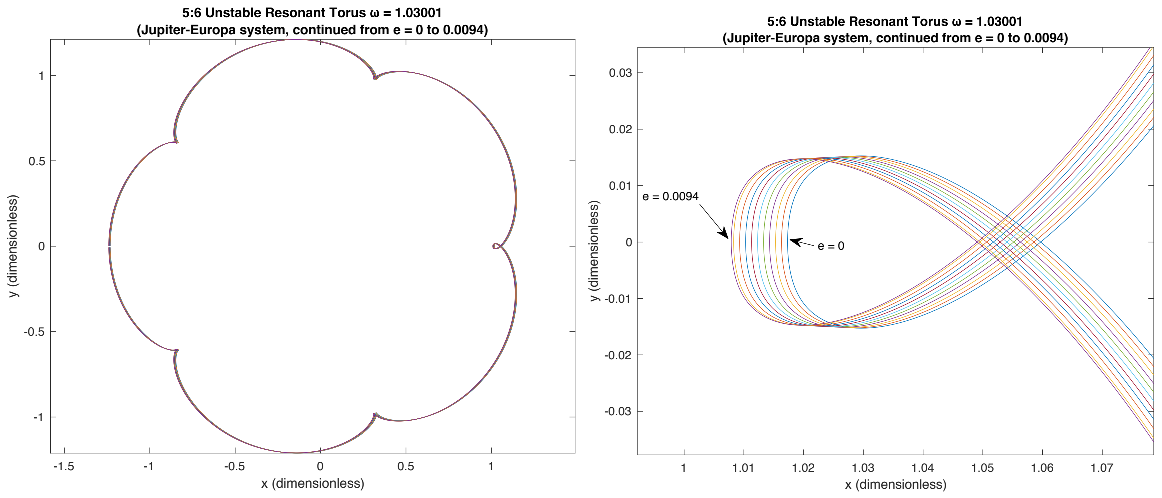


Figure 3.2: Continuation of 5:6 Jupiter-Europa PERTBP resonant torus from $\varepsilon = 0$ to 0.0094

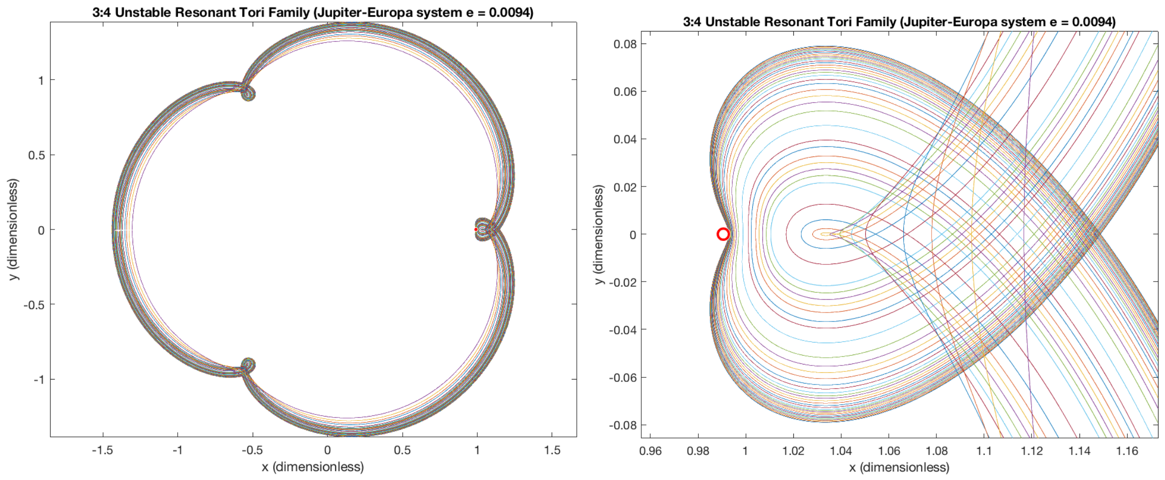


Figure 3.3: Continuation of $\varepsilon = 0.0094$ Jupiter-Europa PERTBP 3:4 resonant tori by ω (Europa surface shown as red circle)

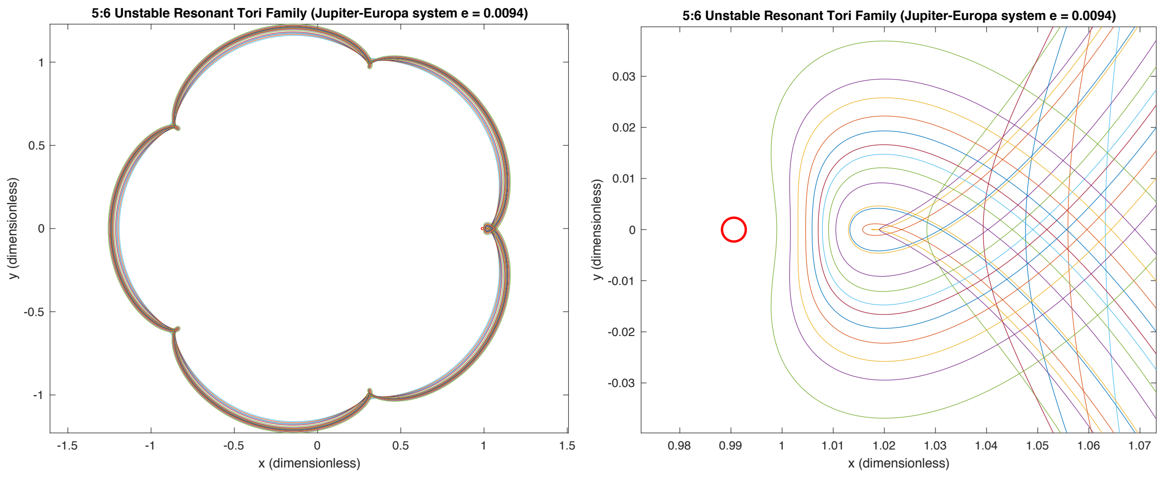
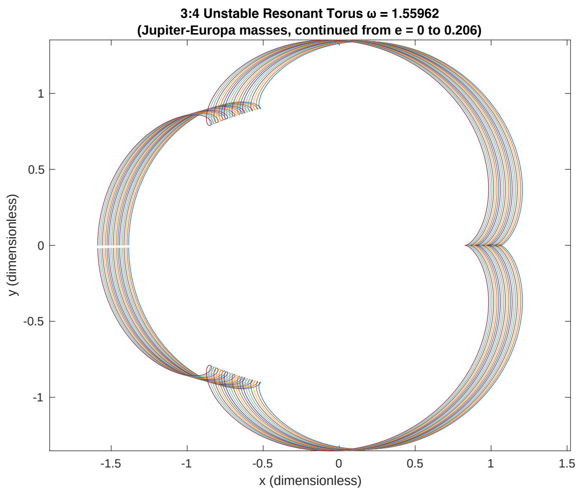


Figure 3.4: Continuation of $\varepsilon = 0.0094$ Jupiter-Europa PERTBP 5:6 resonant tori by ω (Europa surface shown as red circle)

After computing tori in the physical Jupiter-Europa PERTBP with $\varepsilon = 0.0094$, we also tested our quasi-Newton method to see if it would work for larger ε. Fig. 3.5 shows selected tori from the continuation of a 3:4 resonant periodic orbit from the PCRTBP to an invariant circle of the PERTBP with Jupiter-Europa mass ratio μ, but $\varepsilon = 0.206$. This eccentricity is larger than that of the Sun-Mercury system, which has one of the most eccentric two-body orbits of any pair of large solar system bodies. We used $N = 1024$ and a continuation step size of $\Delta\varepsilon = 0.0005$ (the quasi-Newton method failed to converge for larger step sizes); every 20th torus is shown in the figure. As can be seen, our method was robust even for large values of the perturbation ε. On a 2017-era quad-core i7 laptop CPU, our Julia program took only about 230 seconds for the entire continuation to $\varepsilon = 0.206$, and less than 10 s for continuation to the physical value $\varepsilon = 0.0094$.

3.5 Parameterization Method for Stable and Unstable Manifolds

With the invariant circles and their stable and unstable bundles computed, we next turn our attention to accurate computation of torus stable and unstable manifolds. Many studies using manifolds, such as [33], use linear approximations of invariant manifolds found by adding small vectors in the stable or unstable directions to the points of the torus, and



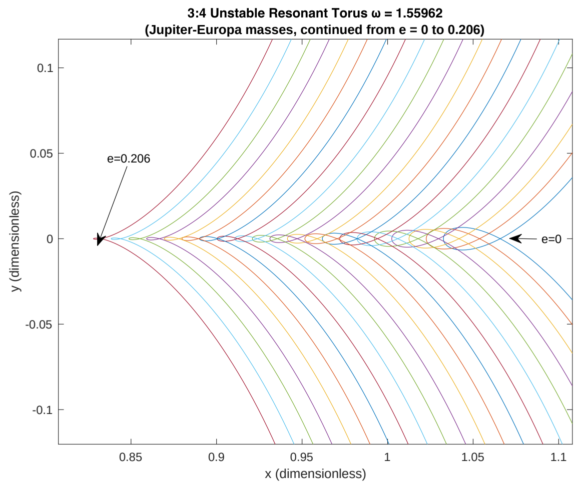


Figure 3.5: Selected tori from 3:4 Jupiter-Europa PERTBP continuation from $\varepsilon = 0$ to 0.206

then integrating backwards or forwards. However, we compute high order Fourier-Taylor polynomials which approximate the manifolds very accurately in some domain of validity. The algorithm used here bears many similarities with the method used in Section 2.4.1 for computation of 1D manifolds of period-maps for periodic orbits in the PCRTBP. A different version of this algorithm was also used by [40] in a lower-dimensional setting.

Since our F-invariant circles are 1D and have one stable and one unstable direction at each point, the circles' stable and unstable manifolds will be 2D and diffeomorphic to either an infinite cylinder or a Möbius strip. A cylinder can be continuously parameterized using an angle θ and a real number s; this actually is also possible for a Möbius strip, as long as the parameterization is non-injective and wraps around the strip twice as θ goes from 0 to 2π (the "double covering" trick we used in Section 3.4.8). In the framework of Section 1.2, we have $M = \mathbb{R}^4$, $\mathcal{M} = \mathbb{T} \times \mathbb{R}$, and $f(\theta, s) = (\theta + \omega, \lambda s)$, where λ is the stable λ_s or unstable λ_u entry of Λ, depending on which manifold we are trying to compute. Without loss of generality, we assume that λ_s and λ_u are constant (see Section 3.4.7). With this, the equation to solve for the parameterization $W : \mathbb{T} \times \mathbb{R} \to \mathbb{R}^4$ of the stable or unstable manifold is

$$F(W(\theta, s)) - W(\theta + \omega, \lambda s) = 0, \quad (\theta, s) \in \mathbb{T} \times \mathbb{R} \tag{3.72}$$

3.5.1 Order-by-Order Method to find W

We express W as a Fourier-Taylor series of form

$$W(\theta, s) = \sum_{k\geq 0} W_k(\theta)s^k = K(\theta) + \sum_{k\geq 1} W_k(\theta)s^k \tag{3.73}$$

where $s = 0$ corresponds to the invariant circle $K(\theta)$ whose manifold we are trying to compute. The s^0 term of W is $K(\theta)$, and the linear term $W_1(\theta)$ is the stable $\mathbf{v}_s(\theta)$ or unstable $\mathbf{v}_u(\theta)$ bundle known from the third or fourth column of P. Hence we need to solve for the higher-order "coefficients" $W_k(\theta) : \mathbb{T} \to \mathbb{R}^4$, $k \geq 2$.

Denote $W_{<k}(\theta, s) = K(\theta) + \sum_{j=1}^{k-1} W_j(\theta)s^j$. Assume we have solved for all $W_j(\theta)$ for $j < k$, so that $F(W_{<k}(\theta, s)) - W_{<k}(\theta + \omega, \lambda s)$ has only s^k and higher order terms. Then, starting with $k = 2$, the recursive method to solve for $W_k(\theta)$ is:

1. Find $E_k(\theta) = [F(W_{<k}(\theta, s)) - W_{<k}(\theta+\omega, \lambda s)]_k$, where $[\cdot]_k$ denotes the s^k coefficient of the term inside brackets. We show how to do this in Section 3.5.2.

2. Find $W_k(\theta)$ such that $W_{<k}(\theta, s) + W_k(\theta)s^k$ cancels $E_k(\theta)s^k$ in Eq. (3.72), thus satisfying Eq. (3.72) up to order s^k. The equation to solve for $W_k(\theta)$ is

$$DF(K(\theta))W_k(\theta) - \lambda^k W_k(\theta + \omega) = -E_k(\theta) \tag{3.74}$$

To solve this, let $W_{k,0} = 0$ and iterate the following sequence to convergence:

$$W_{k,i+1}(\theta) = \begin{cases} \lambda^k DF(K(\theta))^{-1} W_{k,i}(\theta + \omega) - DF(K(\theta))^{-1} E_k(\theta) & \text{if } |\lambda| < 1 \\ \lambda^{-k} DF(K(\theta - \omega)) W_{k,i}(\theta - \omega) + \lambda^{-k} E_k(\theta - \omega) & \text{if } |\lambda| > 1 \end{cases} \tag{3.75}$$

(Fourier methods are an alternate method of solving Eq. (3.74); see Remark 3.5.1)

3. Set $W_{<k+1}(\theta, s) = W_{<k}(\theta, s) + W_k(\theta)s^k$ and return to step 1.

The recursion is stopped when we are satisfied with the degree k of W. We now prove that the equations and method described in Step 2 to find W_k are valid.

Claim. *If W_k solves Eq.* (3.74)*, then for $j \leq k$ (using the $[\cdot]_k$ notation defined earlier),*

$$\left[F(W_{<k}(\theta,s)+W_k(\theta)s^k)-\left(W_{<k}(\theta+\omega,\lambda s)+W_k(\theta+\omega)(\lambda s)^k\right)\right]_j=0 \tag{3.76}$$

Proof. Recall that $F(W_{<k}(\theta,s))-W_{<k}(\theta+\omega,\lambda s)=E_k(\theta)s^k+\mathcal{O}(s^{k+1})$. Expanding Eq. (3.76) in Taylor series and keeping only s^k and lower order terms gives

$$\begin{aligned}&\Big[F(W_{<k}(\theta,s))+DF(W_{<k}(\theta,s))W_k(\theta)s^k-\\ &\qquad\left(W_{<k}(\theta+\omega,\lambda s)+W_k(\theta+\omega)(\lambda s)^k\right)\Big]_j\\ &=[E_k(\theta)s^k+DF(W_{<k}(\theta,s))W_k(\theta)s^k-\lambda^kW_k(\theta+\omega)s^k]_j \\ &=\begin{cases}0 & \text{if } j<k,\\ E_k(\theta)+DF(K(\theta))W_k(\theta)-\lambda^kW_k(\theta+\omega)=0 & \text{if } j=k\end{cases}\end{aligned} \tag{3.77}$$

where the $j=k$ case of the last line follows from the preceding line by dividing s^k out from the quantity inside $[.]_j$, and then taking $s\to 0$. □ □

Claim. *The sequence $\{W_{k,i}\}_{i\in\mathbb{N}}$ defined by $W_{k,0}=0$ and Eq.* (3.75) *converges to W_k.*

$$W_{k,i+1}(\theta)=\begin{cases}\lambda^kDF(K(\theta))^{-1}W_{k,i}(\theta+\omega)-DF(K(\theta))^{-1}E_k(\theta) & \text{if } |\lambda|<1\\ \lambda^{-k}DF(K(\theta-\omega))W_{k,i}(\theta-\omega)+\lambda^{-k}E_k(\theta-\omega) & \text{if } |\lambda|>1\end{cases} \tag{3.78}$$

Proof. Let P,Λ be the bundle and Floquet matrices for $K(\theta)$. We assume that Λ is constant (as the procedure from Section 3.4.7 gives). Then, it is easy to show that

$$W_{k,i}(\theta)=\begin{cases}-P(\theta)\sum_{j=0}^{i-1}\lambda^{kj}\Lambda^{-j-1}[P^{-1}(\theta+(j+1)\omega)E_k(\theta+j\omega)] & \text{if } |\lambda|<1\\ P(\theta)\sum_{j=0}^{i-1}\lambda^{-k(j+1)}\Lambda^{j}[P^{-1}(\theta-j\omega)E_k(\theta-(j+1)\omega)] & \text{if } |\lambda|>1\end{cases} \tag{3.79}$$

solves Eq. (3.75) with $W_{k,0} = 0$; simply substitute Eq. (3.79) for $W_{k,i}$ in Eq. (3.75) and use $DF(K(\theta - \omega))P(\theta - \omega) = P(\theta)\Lambda$ and $DF(K(\theta))^{-1} = P(\theta)\Lambda^{-1}P^{-1}(\theta + \omega)$ to simplify the RHS of the resulting equation.

Now, we will show that $W_k(\theta) = \lim_{i\to\infty} W_{k,i}(\theta)$. First of all, note that

$$\lambda_s^j\Lambda^{-j} = \begin{bmatrix} \lambda_s^j & -j\lambda_s^jT & 0 & 0 \\ 0 & \lambda_s^j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \lambda_s^j\lambda_u^{-j} \end{bmatrix} \qquad \lambda_u^{-j}\Lambda^j = \begin{bmatrix} \lambda_u^{-j} & j\lambda_u^{-j}T & 0 & 0 \\ 0 & \lambda_u^{-j} & 0 & 0 \\ 0 & 0 & \lambda_s^j\lambda_u^{-j} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.80}$$

for all $j \in \mathbb{N}$, where Λ is of the form given in Eq. (3.6) and has constant T, λ_s, and λ_u as assumed earlier. Since $|\lambda_s| < 1$ and $|\lambda_u| > 1$, $\lambda_s^j\Lambda^{-j}$ and $\lambda_u^{-j}\Lambda^j$ are hence bounded for all $j \in \mathbb{N}$. Now, define $\Gamma_s(\theta) = \Lambda^{-1}P^{-1}(\theta + \omega)E_k(\theta)$ and $\Gamma_u(\theta) = \lambda^{-k}P^{-1}(\theta)E_k(\theta - \omega)$; also, recall that $|\lambda| < 1$ means $\lambda = \lambda_s$ and $|\lambda| > 1$ means $\lambda = \lambda_u$. We can use all this to rewrite Eq. (3.79) as

$$W_{k,i}(\theta) = \begin{cases} -P(\theta)\sum_{j=0}^{i-1} \lambda_s^{(k-1)j}[\lambda_s^j\Lambda^{-j}\Gamma_s(\theta + j\omega)] & \text{if } |\lambda| < 1 \\ P(\theta)\sum_{j=0}^{i-1} \lambda_u^{-(k-1)j}[\lambda_u^{-j}\Lambda^j\Gamma_u(\theta - j\omega)] & \text{if } |\lambda| > 1 \end{cases} \tag{3.81}$$

In both $|\lambda| < 1$ and $|\lambda| > 1$ cases of Eq. (3.81), the quantities in square brackets are bounded for all $\theta \in \mathbb{T}$ and $j \in \mathbb{N}$. As $k \geq 2$, $\lambda_s^{k-1} < 1$ and $\lambda_u^{-(k-1)} < 1$; hence, if $i \to \infty$, the sum in Eq. (3.81) is absolutely uniformly convergent. Hence $L(\theta) = \lim_{i\to\infty} W_{k,i}(\theta)$ exists. Letting $i \to \infty$ on both sides of Eq. (3.75) gives

$$L(\theta) = \begin{cases} \lambda^k DF(K(\theta))^{-1}L(\theta + \omega) - DF(K(\theta))^{-1}E_k(\theta) & \text{if } |\lambda| < 1 \\ \lambda^{-k}DF(K(\theta - \omega))L(\theta - \omega) + \lambda^{-k}E_k(\theta - \omega) & \text{if } |\lambda| > 1 \end{cases} \tag{3.82}$$

which for both $|\lambda| < 1$ and $|\lambda| > 1$ is equivalent to Eq. (3.74) with $W_k = L$. □ □

Remark. *Given P and Λ satisfying Eq.* (3.5)*, substituting $W_k = PV_k$ into Eq.* (3.74) *and rearranging gives $\Lambda V_k(\theta) - \lambda^k V_k(\theta+\omega) = -P(\theta+\omega)^{-1}E_k(\theta)$, which can be solved for V_k component by component using the Fourier methods from Section 3.4.4. We used the iteration method of Eq.* (3.75) *instead, to avoid any possible multiplication-translation numerical discretization issues (see Section 3.4.10).*

3.5.2 Computing $E_k(\theta)$: Automatic Differentiation and Jet Transport

In step 1 of the order-by-order method to find W, we compute the s^k coefficient

$$E_k(\theta) = [F(W_{<k}(\theta, s)) - W_{<k}(\theta+\omega, \lambda s)]_k \tag{3.83}$$

In Eq. (3.83), the s^k term of $W_{<k}(\theta+\omega, \lambda s)$ is 0, since $W_{<k}(\theta, s)$ is a Fourier-Taylor series up to order s^{k-1} and λ is constant. However, computing the Fourier-Taylor expansion of $F(W_{<k}(\theta, s))$ is more complicated, as F is a nonlinear stroboscopic map defined by integrating points for a fixed time $2\pi/\Omega_p$ by the equations of motion (1.5) and (1.6). We will need the tools of automatic differentiation [12] and jet transport [26] for this. Note that some researchers [53, 54] use the term differential algebra to refer to what we call automatic differentiation.

Automatic differentiation is an efficient and recursive technique for evaluating operations on Taylor series. For instance, let $f(s)$ and $g(s)$, $s \in \mathbb{R}$, be two series; we can use their known coefficients to compute $d(s) = f(s)/g(s)$ as a Taylor series as well. Let subscript j denote the s^j coefficient of a series; since $f(s) = d(s)g(s)$, we find that $f_i = \sum_{j=0}^{i} d_j g_{i-j} = \left(\sum_{j=0}^{i-1} d_j g_{i-j}(s)\right) + d_i g_0$, which implies that

$$d_i = \frac{1}{g_0}\left(f_i - \sum_{j=0}^{i-1} d_j g_{i-j}\right) \tag{3.84}$$

Starting with $d_0 = f_0/g_0$, Eq. (3.84) allows us to recursively compute d_i, $i \geq 1$. Similar formulas also exist for recursively evaluating many other functions and operations on

Taylor series, including $f(s)^\alpha$, $\alpha \in \mathbb{R}$; see [12] for more examples. Most importantly, in all automatic differentiation formulas, the output series s^i coefficient is a function of only the s^i and lower order coefficients of the input series. Hence, we can use truncated Taylor series with these algorithms when implementing them in computer programs.

Recall from Section 3.4.10 that on the computer, we represent all functions of θ, including the $W_j(\theta)$, as arrays of function values on an evenly spaced grid of θ values $\theta_i = 2\pi i/N$, $i = 0, \dots, N-1$. Note that for fixed θ_i, $W_{<k}(\theta_i, s)$ is a Taylor series (not Fourier-Taylor) with coefficients $W_j(\theta_i) \in \mathbb{R}^4$. Using automatic differentiation, we can substitute Taylor series such as $W_{<k}(\theta_i, s)$ for (x, y, p_x, p_y) in the equations of motion (1.5), which gives us series in s for $(\dot{x}, \dot{y}, \dot{p}_x, \dot{p}_y)$. In terms of computer programming, this means that after overloading the required operators (usually arithmetic and power) to accept Taylor series arguments, we can use numerical integration routines with the series as well.

To be more clear, consider a Taylor series-valued function of time $V(s,t) = \sum_{j=0}^{\infty} V_j(t)s^j : \mathbb{R}^2 \to \mathbb{R}^4$, where $V_j(t)$ are its time-varying Taylor coefficients. Write $V_x(s,t)$, $V_y(s,t)$, $V_{p_x}(s,t)$, and $V_{p_y}(s,t)$ for the x, y, p_x, and p_y components of $V(s,t)$; similarly write $V_{j,x}(t)$, $V_{j,y}(t)$, $V_{j,p_x}(t)$, and $V_{j,p_y}(t)$ for the components of $V_j(t)$. Substituting V in Eq. (1.5) yields a system of differential equations

$$\frac{d}{dt}V_x(s,t) = \sum_{j=0}^{\infty} \dot{V}_{j,x}(t)s^j = \frac{\partial H_\varepsilon}{\partial p_x}\Big(V_x(s,t), V_y(s,t), V_{p_x}(s,t), V_{p_y}(s,t), \theta_p\Big) \tag{3.85}$$

$$\frac{d}{dt}V_y(s,t) = \sum_{j=0}^{\infty} \dot{V}_{j,y}(t)s^j = \frac{\partial H_\varepsilon}{\partial p_y}\Big(V_x(s,t), V_y(s,t), V_{p_x}(s,t), V_{p_y}(s,t), \theta_p\Big) \tag{3.86}$$

$$\frac{d}{dt}V_{p_x}(s,t) = \sum_{j=0}^{\infty} \dot{V}_{j,p_x}(t)s^j = -\frac{\partial H_\varepsilon}{\partial x}\Big(V_x(s,t), V_y(s,t), V_{p_x}(s,t), V_{p_y}(s,t), \theta_p\Big) \tag{3.87}$$

$$\frac{d}{dt}V_{p_y}(s,t) = \sum_{j=0}^{\infty} \dot{V}_{j,p_y}(t)s^j = -\frac{\partial H_\varepsilon}{\partial y}\Big(V_x(s,t), V_y(s,t), V_{p_x}(s,t), V_{p_y}(s,t), \theta_p\Big) \tag{3.88}$$

$$\dot{\theta}_p = \Omega_p \tag{3.89}$$

H_ε and its partials are algebraic functions that are suitable for use with automatic differentiation techniques; see, for instance, the PERTBP Hamiltonian Eq. (1.7). Hence, if the $V_{j,x}(t)$, $V_{j,y}(t)$, $V_{j,p_x}(t)$, $V_{j,p_y}(t)$, and θ_p are known for $j \in \mathbb{N}$ and some $t \in \mathbb{R}$, automatic differentiation allows us to simplify the RHS of each of Eq. (3.85)-(3.88) to a series in s. Then, for each of Eq. (3.85)-(3.88) and $j \in \mathbb{N}$, the s^j coefficient $\dot{V}_{j,x}(t)$, $\dot{V}_{j,y}(t)$, $\dot{V}_{j,p_x}(t)$, or $\dot{V}_{j,p_y}(t)$ from the LHS must be equal to the s^j coefficient of the RHS. In other words, $\dot{V}_{j,x}(t)$, $\dot{V}_{j,y}(t)$, $\dot{V}_{j,p_x}(t)$, and $\dot{V}_{j,p_y}(t)$, $j \in \mathbb{N}$, are functions of θ_p, $V_{j,x}(t)$, $V_{j,y}(t)$, $V_{j,p_x}(t)$, and $V_{j,p_y}(t)$, $j \in \mathbb{N}$. This is effectively a system of differential equations for the time-varying Taylor coefficients of $V(s,t)$. Solving Eq. (3.85)-(3.89) with initial condition $V(s,0) = W_{<k}(\theta_i, s)$ and initial θ_p equal to the value fixed in Section 3.3.1, we can compute $F(W_{<k}(\theta_i, s)) = V(s, 2\pi/\Omega_p)$.

In summary, we consider the Taylor coefficients of $W_{<k}(\theta_i, s)$ as initial state variables to be numerically integrated coefficient by coefficient; propagating by time $2\pi/\Omega_p$, we get the Taylor coefficients of $F(W_{<k}(\theta_i, s))$. Doing this for each $i = 0, \dots, N-1$ is enough to represent the Fourier-Taylor coefficients of $F(W_{<k}(\theta, s))$ on the computer, up to order k; the s^k coefficient of this gives us $E_k(\theta)$. This approach for numerical integration of Taylor series is called jet transport; see [26] for more details. Truncated Taylor series can be used with jet transport, since the automatic differentiation techniques used to evaluate time derivatives work with truncated series. Note that for an n-dimensional state ($n = 4$ in our case) and degree-d truncated series, there are $n(d+1)$ coefficients, which is the required dimension for the numerical integration.

3.5.3 Notes About Numerical Computation of Manifolds

We implemented the parameterization method, automatic differentiation, and jet transport of Sections 3.5.1 and 3.5.2 in a C program for computation of stable and unstable manifolds. For numerical integration, including jet transport, we used the Runge-Kutta Prince-Dormand (8,9) integrator from the GSL library [55]; integrations were parallelized using

OpenMP with one thread for each θ_i value. We tested our tools by computing manifolds of some of the 3:4 and 5:6 Jupiter-Europa PERTBP tori shown in Fig. 3.2 and 3.3, with N ranging from 1024 to 2048. On a quad-core Intel i7 laptop CPU, the program took less than 10 seconds for the computation of s^5-order parameterizations.

Note that in each step of order k, when $F(W_{<k}(\theta_i, s)) - W_{<k}(\theta_i + \omega, \lambda s)$ is computed in order to find $E_k(\theta_i)$, the s^j coefficients for $j < k$ should be zero (to compute the $W_{<k}(\theta_i + \omega, \lambda s)$ coefficients, use the translation algorithm from Section 3.4.10 on the arrays of $W_j(\theta_i)$ values, and then multiply $W_j(\theta_i + \omega)$ by λ^j). This behavior was indeed observed when running the program, serving as a check on the accuracy of the computation. The final s^d-degree truncated series $W_{\leq d}(\theta, s) = K(\theta) + \sum_{j=1}^{d} W_j(\theta)s^j$ satisfies $F(W_{\leq d}(\theta_i, s)) - W_{\leq d}(\theta_i + \omega, \lambda s) = 0$ up to terms of order s^d, for each $i = 0, \dots, N-1$.

In the $W_k(\theta)$ step, we truncate all series at s^k for the automatic differentiation and jet transport steps; this optimizes computational time and storage requirements. Also, note that given $W(\theta, s)$ solving Eq. (3.72), $W(\theta, \alpha s)$ is also a solution for any $\alpha \in \mathbb{R}$. Sometimes, the jet transport integration may struggle to converge as a result of fast-growing coefficients $W_j(\theta)$ of $W(\theta, s)$; in this case, scaling $W(\theta, s)$ to $W(\theta, \alpha s)$ with $\alpha < 1$ can help. To do this, simply multiply $W_1(\theta)$ by α and then restart the order-by-order parameterization method algorithm.

As a final remark, note that in certain systems, such as the PERTBP with $\theta_p = 0$ at $t = 0$, the equations of motion have the same time-reversal symmetry as the PCRTBP. In this case, knowledge of the stable manifold $W^s(\theta, s)$ gives us the unstable manifold $W^u(\theta, s)$ simply by setting $W^u(\theta, s) = MW^s(2\pi - \theta, s)$ where M is the diagonal matrix with diagonal entries $1, -1, -1$, and 1. By doing this, we save half the computation time as compared to computing both W^s and W^u.

3.5.4 Fundamental Domains of Parameterizations

The d degree Fourier-Taylor parameterization $W_{\leq d}(\theta, s)$ of the manifolds of $K(\theta)$ will be more accurate than linear approximation by the stable or unstable direction at each point $K(\theta)$, However, due to series truncation error, $W_{\leq d}(\theta, s)$ is not exact. Furthermore, even the exact infinite series $W(\theta, s)$ satisfying Eq. (3.72) would only be valid for s within some radius of convergence. Thus, we must determine the values of $s \in \mathbb{R}$ for which $W_{\leq d}(\theta, s)$ accurately represents the invariant manifold.

Fix an error tolerance, say $E_{tol} = 10^{-5}$ or 10^{-6}. We now find what [12] call the fundamental domain of $W_{\leq d}(\theta, s)$; this is the largest set $\mathbb{T} \times (-D, D)$ such that for all $(\theta, s) \in \mathbb{T} \times (-D, D)$, the error in invariance Eq. (3.72) is less than E_{tol}. That is, we seek the largest $D \in \mathbb{R}^+$ such that for all s satisfying $|s| < D$,

$$\max_{\theta \in \mathbb{T}} \|F(W_{\leq d}(\theta, s)) - W_{\leq d}(\theta + \omega, \lambda s)\| < E_{tol} \tag{3.90}$$

In practice, since we know $K(\theta)$ and $W_j(\theta)$, $j = 1, \ldots, d$, at the values θ_i, $i = 0, \ldots, N-1$, we search for the largest $D \in \mathbb{R}^+$ such that for all s with $|s| < D$,

$$\max_{i=0,\ldots,N-1} \|F(W_{\leq d}(\theta_i, s)) - W_{\leq d}(\theta_i + \omega, \lambda s)\| < E_{tol} \tag{3.91}$$

The simplest way of finding D is to first use bisection to find the largest D_i such that $\|F(W_{\leq d}(\theta_i, s)) - W_{\leq d}(\theta_i + \omega, \lambda s)\| < E_{tol}$ for all $s \in (-D_i, D_i)$. After doing this for $i = 0, \ldots, N-1$, D will be the minimum of all the D_i.

We computed the fundamental domains of validity for 5 different 3:4 and 5:6 PERTBP resonant torus manifold parameterizations. We found that the domains for $d = 5$ were 50-200 times larger than those for $d = 1$. For linear parameterizations ($d = 1$), the domain size D of all test cases was on the order of 10^{-4} at best, generally 10^{-5}. However, for the degree-5 parameterizations $W_{d \leq 5}(\theta, s)$, D was on the order of 10^{-3} or 0.01. Higher degree parameterizations may improve even further. Note that a larger fundamental domain means that less numerical integration is required for manifold globalization, reducing the computation time.

3.6 Globalization, Regularization, and Visualization

At this point, we have accurate local representations of stable and unstable manifolds of our stroboscopic map invariant circles. Given a manifold's Fourier-Taylor parameterization $W_p(\theta, s)$ and its fundamental domain $\mathcal{D} = \mathbb{T} \times (-D, D)$, the image $W_p(\mathcal{D})$ gives us a piece of the manifold in the map phase space $\mathbb{R}^4$. However, this subset of the manifold will be close to its base invariant circle $K(\theta)$; generally, it is motions on the manifold further away from the torus that are of interest for applications. Hence, we need to extend our Fourier-Taylor parameterization $W_p : \mathcal{D} \to \mathbb{R}^4$ to a function $W : \mathbb{T} \times \mathbb{R} \to \mathbb{R}^4$ parameterizing the entire manifold, with $W = W_p$ on $\mathcal{D}$. This is referred to as globalization.

Recall from Eq. (3.72) that W must satisfy $F(W(\theta, s)) = W(\theta + \omega, \lambda s)$. Applying this repeatedly, we have that $F^k(W(\theta, s)) = W(\theta + k\omega, \lambda^k s)$, where the superscript $k \in \mathbb{Z}^+$ refers to function composition. We can rewrite this as

$$W(\theta, s) = F^k(W(\theta - k\omega, \lambda^{-k} s)) \tag{3.92}$$

$$W(\theta, s) = F^{-k}(W(\theta + k\omega, \lambda^k s)) \tag{3.93}$$

Eq. (3.92)-(3.93) allow us to define $W(\theta, s)$ for all $(\theta, s) \in \mathbb{T} \times \mathbb{R}$. If W is an unstable manifold with $|\lambda| > 1$, choose $k \geq 0$ such that $|\lambda^{-k} s| < D$ and use W_p to evaluate Eq. (3.92); if W is a stable manifold with $|\lambda| < 1$, take $k \geq 0$ such that $|\lambda^k s| < D$ and evaluate Eq. (3.93). The map F^k (or F^{-k}) is just a time $2\pi k/\Omega_p$ (or $-2\pi k/\Omega_p$) numerical integration. $W(\theta, s)$ thus defined satisfies $F(W(\theta, s)) = W(\theta + \omega, \lambda s)$ for all $(\theta, s) \in \mathbb{T} \times \mathbb{R}$, so the image $W(\mathbb{T} \times \mathbb{R})$ is F-invariant. Hence, Eq. (3.92)-(3.93) give us a global representation of the entire stable or unstable manifold. Note that W can be differentiated easily with respect to θ and s to get the tangent vectors to the manifold, as $DF^{\pm k}$ is a state transition matrix and DW_p only requires polynomial or Fourier series differentiation. This will be used in Section **??** of the following chapter for differential correction of approximate heteroclinic connections.

3.6.1 Mesh Representations of Globalized Manifolds

For visualization, we often want to calculate a mesh of many points on the manifold, rather than just a few $W(\theta, s)$ values. To do this, we first take an evenly-spaced grid of L s-values $\{s_j\}$ from $-D$ to D, in addition to our grid of N θ values θ_i, and then directly evaluate the Fourier-Taylor parameterization to compute $W(\theta_i, s_j)$ for each $i = 0, \dots, N-1$, $j = 1, \dots, L$. Next, we repeatedly apply F or F^{-1} to the $W(\theta_i, s_j)$ to get the points $W(\theta_i + k\omega, \lambda^k s_j) = F^k(W(\theta_i, s_j))$ if $|\lambda| > 1$ or $W(\theta_i - k\omega, \lambda^{-k} s_j) = F^{-k}(W(\theta_i, s_j))$ if $|\lambda| < 1$, for $k = 0, 1, 2, \dots$ up to some $k_{max} \in \mathbb{Z}^+$. The numerical integrations required in this step may require use of regularized equations of motion, which we will discuss in Section 3.6.2. Finally, we use the translation algorithm given in Eq. (3.68) to find the points $W(\theta_i, \lambda^k s_j)$ if $|\lambda| > 1$ or $W(\theta_i, \lambda^{-k} s_j)$ if $|\lambda| < 1$; we need all points to be at the same set of N θ_i values when trying to create a manifold mesh that can be plotted.

By following this procedure, we get a discretized, plottable representation of the manifold subset $\{W(\theta, s) : (\theta, s) \in \mathbb{T} \times [-M, M]\}$, where $M = \lambda^{k_{max}} D$ if $|\lambda| > 1$ and $M = \lambda^{-k_{max}} D$ if $|\lambda| < 1$. Note that the numerical integrations can be parallelized across θ_i values, which we took advantage of. A 3D projection of an example globalized stable manifold mesh (denoted W^s) of a 3:4 Jupiter-Europa PERTBP invariant circle is given in Fig. 3.6, with $N = 1024$, $L = 101$, $k_{max} = 6$.

3.6.2 The Need for Regularization: An Extension of Levi-Civita to the PERTBP

In the equations of motion for the PERTBP and other periodically-perturbed PCRTBP models, the positions of the two large masses m_1 and m_2 are singularities. However, when

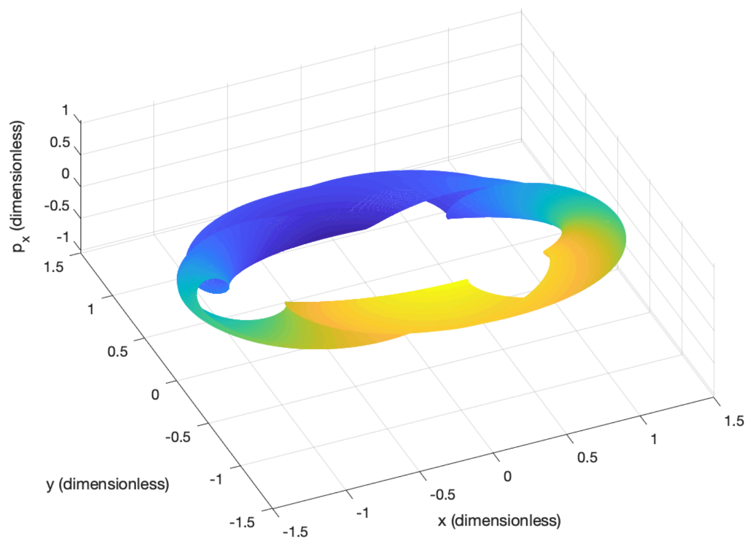


Figure 3.6: (x, y, p_x) projection of Jupiter-Europa PERTBP 3:4 W^s for $\omega = 1.559620297$

numerically integrating points forwards or backwards during the manifold mesh computation described in Section 3.6.1, it is possible for some points' trajectories to pass extremely close to the singularity at m_2. Moreover, this can indicate that the manifold being computed actually passes through the m_2 singularity. Such behavior was observed, for example, during computation of manifold meshes for 5:6 Jupiter-Europa PERTBP tori. These close approaches to m_2 can result in numerical issues, including lack of integrator convergence.

In the PCRTBP, the Levi-Civita regularization is very commonly used to compute trajectories which pass near or through a singularity; see [5] for full details. First, a canonical coordinate transformation is applied to the PCRTBP Hamiltonian H_0 from Eq. (1.4). This is followed by the addition of a pair of action-angle variables to the transformed Hamiltonian; the new action's value is set to $-H_0$, which has a constant value along the trajectory. This finally allows a time-rescaling to be used which cancels the singularity. This method, however, relies on the fact that H_0 is constant along PCRTBP trajectories. For our periodically-perturbed models, this is not the case. Hence, some modification is required.

For the PERTBP, the singularity corresponding to m_2 is the time-varying point $(x, y) = ((1-\mu)(1-\varepsilon\cos E(t)), 0)$. We now present the derivation of the modified Levi-Civita regularization of m_2 for the PERTBP; we expect very similar methods to apply for other periodically-perturbed PCRTBP models as well. Readers primarily interested in using the final regularized equations for numerical integration should skip to Section 3.6.2. The following is heavily inspired by [5].

First, take the PERTBP Hamiltonian H_ε from Eq. (1.7) and add a momentum variable p_t conjugated to t. The Hamiltonian and equations of motion become

$$\bar{H}_\varepsilon(p_x, p_y, p_t, x, y, t) = p_t + \frac{p_x^2 + p_y^2}{2} + n(t)(p_x y - p_y x) - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{3.94}$$

$$\dot{x} = \frac{\partial \bar{H}_\varepsilon}{\partial p_x} \quad \dot{y} = \frac{\partial \bar{H}_\varepsilon}{\partial p_y} \quad \dot{t} = \frac{\partial \bar{H}_\varepsilon}{\partial p_t} \qquad \dot{p}_x = -\frac{\partial \bar{H}_\varepsilon}{\partial x} \quad \dot{p}_y = -\frac{\partial \bar{H}_\varepsilon}{\partial y} \quad \dot{p}_t = -\frac{\partial \bar{H}_\varepsilon}{\partial t} \tag{3.95}$$

where $r_1 = \sqrt{\left(x + \mu(1+\chi(t))\right)^2 + y^2}$, $r_2 = \sqrt{\left(x - (1-\mu)(1+\chi(t))\right)^2 + y^2}$, and $\chi(t) = -\varepsilon \cos E(t)$. Note that adding p_t does not change the values of $\dot{x}$, $\dot{y}$, $\dot{p}_x$, $\dot{p}_y$, and $\dot{t} = \partial \bar{H}_\varepsilon / \partial p_t = 1$ as compared to using Eq. (1.7). However, unlike H_ε, the new Hamiltonian $\bar{H}_\varepsilon$ does remain constant along trajectories in (p_x, p_y, p_t, x, y, t) space. Also, given an initial condition (x, y, p_x, p_y, t) to be propagated, the initial value of p_t should be set so that $\bar{H}_\varepsilon = 0$; this will be important later on.

Now, we perform a canonical coordinate transformation. This is required in order to "straighten out" certain trajectories passing through the singularity which make sharp bends in physical space [56]. Define a generating function

$$W(p_x, p_y, p_t, X, Y, T) = p_x\left(X^2 - Y^2 + (1-\mu)(1+\chi(T))\right) + p_2(2XY) + p_t T \tag{3.96}$$

which is a function of the old momenta and new configuration space coordinates. Then, this defines a transformation between the old (p_x, p_y, p_t, x, y, t) variables and new (P_X, P_Y, P_T, X, Y, T)

variables through the relations [51]

$$
\begin{aligned}
x &= \frac{\partial W}{\partial p_x} = X^2 - Y^2 + (1-\mu)(1+\chi(T)) \qquad & y &= \frac{\partial W}{\partial p_y} = 2XY \\
P_X &= \frac{\partial W}{\partial X} = 2p_x X + 2p_y Y & P_Y &= \frac{\partial W}{\partial Y} = -2p_x Y + 2p_y X \\
t &= \frac{\partial W}{\partial p_t} = T & P_T &= \frac{\partial W}{\partial T} = (1-\mu)p_x \frac{d\chi}{dt}(T) + p_t
\end{aligned}
\tag{3.97}
$$

Eq. (3.97) gives us x, y, and t in terms of the new variables. We can also solve for p_x and p_y to get $p_x = \frac{2}{R}(P_X X - P_Y Y)$ and $p_y = \frac{2}{R}(P_X Y + P_Y X)$, where $R = 4(X^2 + Y^2)$. This then gives us $p_t = P_T - \frac{2}{R}(1-\mu)(P_X X - P_Y Y)\frac{d\chi}{dt}(T)$. Note that in the new variables, $r_2 = \sqrt{(X^2 - Y^2)^2 + (2XY)^2} = R/4$.

Substituting the previous expressions for (p_x, p_y, p_t, x, y, t) into Eq. (3.94) gives

$$
\begin{aligned}
\mathcal{H}_\varepsilon(P_X,&P_Y, P_T, X, Y, T) = \\
&P_T - \frac{2}{R}(1-\mu)(P_X X - P_Y Y)\frac{d\chi}{dt}(T) + \frac{P_X^2 + P_Y^2}{2R} \\
&+ 2n(T)\left[\frac{1}{4}(P_X X - P_Y Y) - \frac{(1-\mu)}{R}(1+\chi(T))(P_X Y + P_Y X)\right] \\
&- \frac{1-\mu}{\sqrt{(X^2+Y^2)^2 + (1+\chi(T))^2 + 2(X^2 - Y^2)(1+\chi(T))}} - \frac{4\mu}{R}
\end{aligned}
\tag{3.98}
$$

The m_2 singularity is now at $(X, Y) = (0, 0)$, where $R = 4r_2 = 0$. Since this was a canonical transformation, the equations of motion in the new coordinates will be

$$
\dot{X} = \frac{\partial \mathcal{H}_\varepsilon}{\partial P_X} \;\; \dot{Y} = \frac{\partial \mathcal{H}_\varepsilon}{\partial P_Y} \;\; \dot{T} = \frac{\partial \mathcal{H}_\varepsilon}{\partial P_T} \quad \dot{P}_X = -\frac{\partial \mathcal{H}_\varepsilon}{\partial X} \;\; \dot{P}_Y = -\frac{\partial \mathcal{H}_\varepsilon}{\partial Y} \;\; \dot{P}_T = -\frac{\partial \mathcal{H}_\varepsilon}{\partial T} \tag{3.99}
$$

To regularize the singularity at $R = 0$, we want to be able to use $R\mathcal{H}_\varepsilon$ instead of $\mathcal{H}_\varepsilon$. For this, define a rescaled time s such that $dt = R\,ds$. Then, we have that $\frac{d}{ds} = \frac{d}{dt}\frac{dt}{ds} = R\frac{d}{dt}$.

Thus, letting prime ($'$) denote d/ds,

$$\begin{aligned} X' &= R\frac{\partial \mathcal{H}_\varepsilon}{\partial P_X} \quad Y' = R\frac{\partial \mathcal{H}_\varepsilon}{\partial P_Y} \quad T' = R\frac{\partial \mathcal{H}_\varepsilon}{\partial P_T} \\ P_X' &= -R\frac{\partial \mathcal{H}_\varepsilon}{\partial X} \quad P_Y' = -R\frac{\partial \mathcal{H}_\varepsilon}{\partial Y} \quad P_T' = -R\frac{\partial \mathcal{H}_\varepsilon}{\partial T} \end{aligned} \tag{3.100}$$

Since R is a function of only X and Y, it is immediate that $X' = \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_X}$, $Y' = \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_Y}$, $T' = \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_T}$, and $P_T' = -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial T}$. Furthermore, since p_t was chosen earlier to ensure $\bar{H}_\varepsilon = 0$, we also will have $\mathcal{H}_\varepsilon = 0$ along the trajectory in (P_X, P_Y, P_T, X, Y, T) space. Hence, we find that $\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial X} = R\frac{\partial \mathcal{H}_\varepsilon}{\partial X} + \mathcal{H}_\varepsilon\frac{\partial R}{\partial X} = R\frac{\partial \mathcal{H}_\varepsilon}{\partial X}$. This yields $P_X' = -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial X}$; we similarly find $P_Y' = -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial Y}$. As $R\mathcal{H}_\varepsilon$ has no singularity at $R = 0$, we thus obtain the m_2-regularized time-s equations of motion

$$\begin{aligned} X' &= \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_X} \quad Y' = \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_Y} \quad T' = \frac{\partial[R\mathcal{H}_\varepsilon]}{\partial P_T} \\ P_X' &= -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial X} \quad P_Y' = -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial Y} \quad P_T' = -\frac{\partial[R\mathcal{H}_\varepsilon]}{\partial T} \end{aligned} \tag{3.101}$$

Usage of Regularized PERTBP Equations of Motion

Let (x^i, y^i, p_x^i, p_y^i) be an initial state we wish to integrate from $t = t_i$ to t_f in the PERTBP. Recall H_ε from Eq. (1.7), and $\bar{H}_\varepsilon$ from Eq. (3.94), with $\chi(t) = -\varepsilon \cos E(t)$. To use the m_2-regularized equations of motion for this integration, we:

1. Set $p_t^i = -H_\varepsilon(p_x^i, p_y^i, x^i, y^i, t_i)$, so that $\bar{H}_\varepsilon(p_x^i, p_y^i, p_t^i, x^i, y^i, t_i) = 0$.

2. Compute initial $(P_X^i, P_Y^i, P_T^i, X^i, Y^i, T^i)$ using the equations (where $j = \sqrt{-1}$)

$$\begin{aligned} &X + jY = \big(x - (1-\mu)(1+\chi(t)) + jy\big)^{1/2} \\ &P_X = 2p_xX + 2p_yY \qquad P_Y = -2p_xY + 2p_yX \\ &T = t \qquad P_T = (1-\mu)p_x\frac{d\chi}{dt}(t) + p_t \end{aligned} \tag{3.102}$$

The sign chosen during the complex square root for $X + jY$ does not matter.

3. Integrate the initial condition $(P_X^i, P_Y^i, P_T^i, X^i, Y^i, T^i)$ using Eq. (3.101), where $R = 4(X^2+Y^2)$ and $\mathcal{H}_\varepsilon$ is given by Eq. (3.98). Only stop the integration when $T = t_f$; do not stop before this occurs, even if the integration time reaches t_f.

4. Transform the resulting final state back to (p_x, p_y, p_t, x, y, t) coordinates using

$$\begin{aligned} &x = X^2 - Y^2 + (1-\mu)(1+\chi(T)) && y = 2XY \\ &p_x = \frac{2}{R}(P_X X - P_Y Y) && p_y = \frac{2}{R}(P_X Y + P_Y X) \\ &t = T = t_f && p_t = P_T - \frac{2}{R}(1-\mu)(P_X X - P_Y Y)\frac{d\chi}{dt}(T) \end{aligned} \tag{3.103}$$

The first line of Eq. (3.102) should be interpreted as two real equations corresponding to setting real and imaginary parts of both sides equal. It follows from the first line of Eq. (3.97) combined with the relation $(X + jY)^2 = X^2 - Y^2 + j(2XY)$. Also, for step 3 above, the requirement to stop integration when $T = t_f$ can be implemented using the "events" functionality of MATLAB's ODE solvers, or the callback features in Julia's DifferentialEquations.jl library.

The partial derivatives of $R\mathcal{H}_\varepsilon$ appearing in the equations of motion Eq. (3.101) are straightforward to compute but lengthy, so we do not write them here. Throughout the steps listed above, as well as for computing the partial derivatives, we need the quantities $\frac{d\chi}{dt}$, $\frac{d^2\chi}{dt^2}$, and $\frac{dn}{dt}$. These are given by the equations

$$\frac{d\chi}{dt} = \frac{\varepsilon \sin E}{1-\varepsilon\cos E} \qquad \frac{d^2\chi}{dt^2} = \frac{\varepsilon\cos E - \varepsilon^2}{(1-\varepsilon\cos E)^3} \qquad \frac{dn}{dt} = \frac{-2\varepsilon\sqrt{1-\varepsilon^2}}{(1-\varepsilon\cos E)^4}\sin E$$

which can be derived from $\chi(t) = -\varepsilon\cos E(t)$, $n(t) = \frac{\sqrt{1-\varepsilon^2}}{(1-\varepsilon\cos E(t))^2}$, and the relation $\frac{dE}{dt} = \frac{1}{1-\varepsilon\cos E}$ (which in turn follows from taking the time derivative of the standard Kepler's equation $M = E - \varepsilon\sin E$; see [2]).

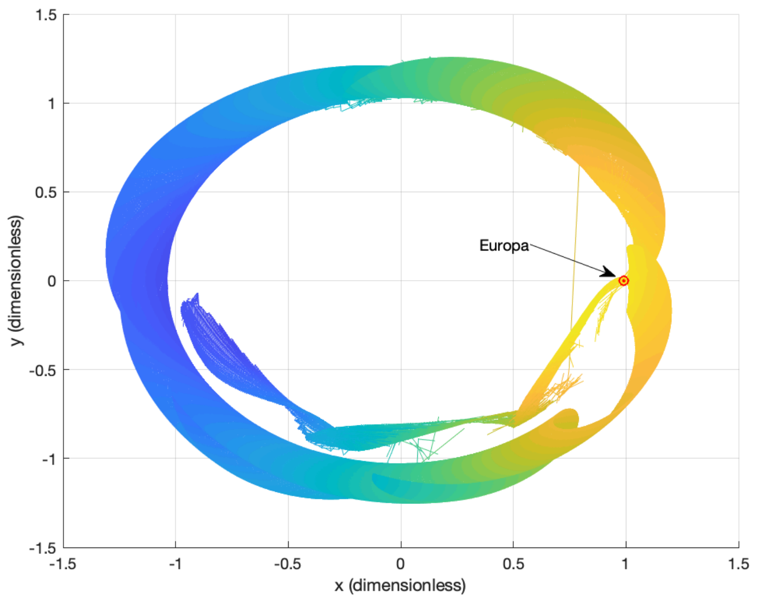


Figure 3.7: (x, y) projection of Jupiter-Europa PERTBP 5:6 W^s for $\omega = 1.030011437$

Computational Results

In Fig. 3.7, we show a 2D projection of an example globalized stable manifold mesh for a 5:6 Jupiter-Europa PERTBP invariant circle. This was computed using the regularized PERTBP equations of motion to evaluate F^{-k} in the procedure described in Section 3.6.1; in this case, $N = 2048$, $L = 101$, $k_{max} = 6$. We have filtered the computed mesh points so as to only plot those which did not result in a very "visually discontinuous" mesh; this filtering is needed during visualization, since close flybys of m_2 can send points which started close together in extremely different directions. Nevertheless, even after discarding some mesh points, it is clearly visible in the figure that the manifold passes through the singularity at m_2 (Europa, marked as a red circle). Using the regularized equations, we experienced no warnings of integrator divergence during program runtime, which were encountered when using the unregularized equations.

We also used the regularized equations of motion to recompute the 3:4 W^s manifold mesh shown earlier in Fig. 3.6. The results matched those which were obtained earlier when using Eq. (1.5) and (1.7) for the numerical integrations, thus verifying the correctness of the regularization procedure. We carried out this computation in Julia, using the DP5 integrator and parallelizing across θ_i with the EnsembleProblem feature of DifferentialEquations.jl [57]; the computation of the 3:4 manifold with $N = 1024$, $L = 101$, $k_{max} = 15$ took approximately 250 seconds on the same quad core i7 laptop CPU used earlier.

3.7 Conclusion

In this chapter, we first developed a quasi-Newton method for the simultaneous computation of unstable invariant circles and their symplectic conjugate center, stable, and unstable bundles for stroboscopic maps of periodically-perturbed PCRTBP models. Our method improves the computational complexity of the torus calculation to $O(N \log N)$ as compared to $O(N^3)$ for the methods used in almost all the existing astrodynamics literature, in addition to giving useful information on the torus stable and unstable directions. Our method also extends the $O(N \log N)$ method of [52] and [12] to unstable tori with center directions, as is the case for the vast majority of celestial mechanics applications. We used this quasi-Newton method for continuation of tori and bundles by both perturbation parameter and rotation number, and described how to initialize the continuation from PCRTBP periodic orbits. We also gave a set of numerical best practices to aid in quasi-Newton method convergence.

After finding the tori and bundles, we used the results of the continuation to start an order-by-order method for the computation of Fourier-Taylor series parameterizations of stable and unstable manifolds for the invariant circles. We found significant improvements in accuracy and fundamental domain size compared to linear manifold approximations. Finally, we were able to extend the parameterizations to compute points of the manifolds outside the fundamental domain, with the aid of a modified Levi-Civita regularization which we derived for the PERTBP. We expect that similar methods can be used to regularize other periodically-perturbed PCRTBP models as well.

The tools developed were tested in the Jupiter-Europa PERTBP, with the calculations of the circles, bundles, and manifold parameterizations taking just a few seconds on a 2017-era laptop with a quad-core Intel i7 CPU. Our Julia program for computation of meshes of globalized manifold points took a few minutes for each manifold, due to the large number of numerical integrations involved. As we describe in the following chapter (based on the paper [15]), with the help of modern computer graphics processing units, these manifold parameterizations and meshes can be used to very rapidly search for and accurately compute intersections of stable and unstable manifolds leading to heteroclinic connections. The methods presented in this chapter can form an important component for low-energy mission design and transfer trajectories in such periodically-perturbed PCRTBP models.

CHAPTER 4
USING GPUS AND THE PARAMETERIZATION METHOD FOR RAPID SEARCH AND REFINEMENT OF CONNECTIONS BETWEEN TORI IN PERIODICALLY PERTURBED PCRTBP MODELS

4.1 Introduction

Numerous prior studies have used the stable and unstable manifolds of unstable planar periodic orbits, both around libration points as well as at resonances, as an efficient tool for multi-body mission design in the planar circular restricted 3-body problem (PCRTBP). For instance, Anderson and Lo [16, 17] studied intersections of the manifolds of resonant periodic orbits in the Jupiter-Europa system as a mechanism of resonance transitions. The book of Koon et al. [36] describes the Poincaré section method of finding intersections of manifolds between L1 and L2 libration point orbits, and shows how to use the resulting regions to construct trajectories with arbitrary itineraries between the different realms of the PCRTBP model. As the phase space is 4-dimensional, fixing an energy level restricts the dynamics to a 3D submanifold, and the Poincaré section further reduces the dimensionality of the system to 2D. Since the manifolds of the periodic orbits are 2D cylinders in the full phase space, taking the Poincaré section reduces the manifolds to 1D curves in the section, so the problem of finding connections between periodic orbits reduces to finding the intersection of two 1D manifold curves in a 2D plane (for example, see Chapter 2).

While this method of intersecting 1D curves is useful in the 4-dimensional phase space of the PCRTBP, it is not applicable to systems with higher dimensional phase spaces. For instance, in the 6D phase space of the spatial CRTBP, fixing an energy level and taking a Poincaré section still results in a 5D energy level and 4D section to be explored. In the case of a time-varying periodic perturbation of the PCRTBP, the phase space becomes 5-

dimensional, considering the perturbation phase as an angular state variable. As energy is no longer conserved in non-autonomous systems, taking a Poincaré section again leaves us with a 4D space to be explored. Furthermore, in these systems, unstable periodic orbits are no longer the main dynamical structures of interest, as 2D manifolds of periodic orbits do not generically intersect each other in a 5D phase space or energy level; instead, unstable quasi-periodic orbits and their manifolds are the objects of sufficient dimensionality such that one can expect intersections.

In Chapter 3, we described how most PCRTBP unstable periodic orbits will persist as 2D unstable quasi-periodic orbits in the 5D phase space of periodically-perturbed PCRTBP models. By considering stroboscopic maps instead of the continuous-time flow, we reduced the dimensionality of the system by 1 so that these quasi-periodic orbits become invariant 1D tori (circles) in the 4D stroboscopic map phase space (x, y, p_x, p_y); these invariant circles have 2D cylindrical stable and unstable manifolds as the PCRTBP unstable periodic orbits did. However, due to the absence of an energy integral, manifold intersections in the perturbed system will occur at isolated points, rather than along continuous trajectory curves. Hence, a different method of computing homoclinic and heteroclinic connections in the full 4D phase space is required. The purpose of this study is to develop new and computationally fast methods and tools for this problem.

In this chapter, we start with a brief overview of graphics processing unit (GPU) computing capabilities and paradigms, as well as some background on collision detection methods from computer graphics, both of which are used in this study. Next, we describe the method of layers for restricting the homoclinic and heteroclinic connection search to appropriate subsets of the two manifolds of interest. Once these subsets are identified, we can computationally represent each manifold as a 2D mesh of points in the 4D stroboscopic map phase space. With these meshes representing the manifolds, we next develop a heavily parallel algorithm for detecting and computing intersections of these meshes in 4D space; we then describe how to implement this method using the Julia programming language and OpenCL, taking advantage of the capabilities of modern GPUs to massively speed up the algorithm execution time. Finally, we take the approximate intersections of manifolds computed from the mesh intersection search, and show how to use our manifold parameterizations to refine the intersections to high precision. We demonstrate the use of our methods by applying them to the search for heteroclinic connections between resonances in the Jupiter-Europa planar elliptic RTBP.

4.2 Background

4.2.1 An Overview of GPU Computing

Graphics processing units (GPUs) are special-purpose computer processors originally designed for executing computations required for 3D graphics [58]. More recently, as the capabilities of GPUs have grown, it has become possible to use GPUs for many other computational tasks as well. GPUs excel at tasks with large and highly parallel computational requirements, as the GPU processes blocks of many elements in parallel using the same program. While a single-program multiple-data (SPMD) programming model is supported on GPUs, due to the lock-step execution of the program on multiple data elements, it is necessary to evaluate both sides of any code branches for all elements in a block of data. Hence, GPUs are best suited to straight-line programs which are mostly written in a single-instruction, multiple data (SIMD) style. Flow control should be kept to a minimum.

There are two main toolkits used for programming GPUs directly. The most commonly used is Nvidia CUDA [59], which works only with Nvidia GPUs. The other is OpenCL, [60] which is an open standard which provides a programming language and APIs that can be used with a variety of SIMD-capable devices, including both AMD and Nvidia GPUs as well as CPUs. OpenCL implementations exist for many different platforms, including Windows, MacOS, and Linux x64. The basic programming concepts of CUDA and OpenCL are very similar; as we used OpenCL in this study, we briefly go over the OpenCL model here, covering the concepts used in this work.

In OpenCL, the fundamental task is the programming of kernels. A *kernel* is a program which executes on the OpenCL device, like a GPU. When a kernel is submitted for execution, a 1D, 2D, or 3D space of indices is defined, called an *NDRange* (we only use a 1D NDRange in this study); each index corresponds to a separate execution of the kernel. These kernel instances are referred to as *work items*, with each work item identified by a unique global ID (a nonnegative integer if NDRange is 1D) corresponding to an index in NDRange. Each work item executes the same kernel, but can access different data in memory. Work items are grouped into *work groups*; all work items in a work group execute concurrently. Finally, the work groups taken together form the NDRange. The key advantage of GPUs is the massive number of threads they have, usually on the order of thousands, which allows them to execute large numbers of work items in parallel.

The OpenCL memory model has various categories of memory stored on the device, accessible to different parts of the program. *Private* memory is accessible only to a single work item, and *local* memory is accessible to all work items in the same work group. We do not use local memory in the programs written for this study. Finally, *global* memory is accessible to all work items. The use of global memory can be a problem if two work items try to access the same memory location at the same time; it is for this purpose that atomic functions are useful. These functions receive a pointer to a 32 bit integer or floating point number stored in global or local memory, modify the value there, and return the old value. The key is that atomic functions execute so that when one work item is carrying out an atomic operation at a pointer location, the other threads must wait for that work item to complete the operation. Some such functions are atomic add, subtract, increment, min, and max.

An example of a simple OpenCL kernel is given in Fig. 4.1. This program takes as input an array of zeros and ones (denoted as in_array) and finds the in_array entries which have a value of 1, saving their indices in the array true_idxs. It also takes a pointer curr_idx to an integer which is initialized to 0 before the kernel is run. The first line after the kernel

```
__kernel void findall(__global const char *in_array, __global uint
 *true_idxs, __global int *curr_idx)
    {
        unsigned int gid = get_global_id(0);

        if (in_array[gid] == 1){
            uint old_val = atomic_inc( curr_idx );
            true_idxs[old_val] = gid+1;
        }

    }
```

Figure 4.1: Example of OpenCL findall kernel

declaration stores the work item global ID as gid. Then, the kernel checks if the gid entry of in_array is 1, so that each work item checks a different entry of in_array. If the gid entry is true, an atomic increment is applied to curr_idx, with the pre-increment value of curr_idx stored in old_val; each work item stores its value of old_val in private memory, separately from all other work items. curr_idx is in global memory, so it is shared between all work items. Hence, the first work item to apply atomic_inc sets its private old_val to 0 and the shared curr_idx to 1, the next work item to apply atomic_inc sets its old_val to 1 and curr_idx to 2, and so on. This way, each work item with in_array[gid] equal to 1 gets a unique consecutive value of old_idx. Finally, the gid values with in_array[gid] equal to 1 are stored in the true_idxs array at the indices old_idx by the corresponding work items (we add 1 to gid since we later use the true_idxs array in Julia, which has one-based indexing).

4.2.2 Collision Detection

Given two sets A and B, each containing some finite number of geometric objects, the problem of collision detection simply is that of determining whether any object from A intersects any object(s) from B, and if so, which pairs of objects intersect and where they do so. Computational methods for collision detection are used in a variety of applications, including video game development, animation, robotics, computer aided design, and physics simulations [61, 62]. Indeed, as will be shown in Section 4.4, such methods are also applicable to the problem of finding heteroclinic connections in dynamical systems.

If one were to solve the collision detection problem by checking each object in A against each object in B for an exact determination of whether/where they intersect, this would require $|A||B|$ tests; even if $A = B$ as is often the case in computer graphics, though not in this chapter, there would still be $\frac{|A|(|A|-1)}{2}$ pairwise intersection tests required. Either way, the cost of such a collision detection algorithm is quadratic, which can quickly become infeasible if $|A|$ and $|B|$ are large (as will be the case later in this chapter) and the pairwise intersection test itself is not extremely computationally cheap. However, unless the objects of A and B are all very simple (e.g. spheres), the exact pairwise intersection test is generally expensive.

It is hence necessary to find a way to reduce the number of exact pairwise tests done. To this end, most collision detection algorithms are comprised of two phases: a broad phase and a narrow phase [62]. In practice, most pairs of objects do not intersect; hence, the broad phase uses very cheap computations to eliminate most of these non-intersecting pairs from consideration. Usually this involves dividing objects into groups such that only objects within the same group can intersect, and/or putting pairs of objects through a very simple test which, if failed, verifies non-intersection. The broad phase outputs a set of potentially colliding pairs of objects, to which the narrow phase then applies the more costly exact intersection test.

4.3 The Method of Layers for Restricting the Connection Search

After using the methods of Chapter 3 to compute the stable/unstable manifolds of unstable invariant tori (invariant circles) for a periodically-perturbed PCRTBP stroboscopic map, we wish to search for heteroclinic connections between them. Henceforth, let $W_1^u(\theta_u, s_u)$ and $W_2^s(\theta_s, s_s)$ be functions parameterizing the unstable and stable manifolds of stroboscopic map-invariant circles 1 and 2, respectively, in the same vein as Sections 3.5-3.6. Heteroclinic connections from circle 1 to circle 2 occur when the images of W_1^u and W_2^s intersect in (x, y, p_x, p_y) space. This means we need to find (θ_u, s_u) and (θ_s, s_s) such that

$$W_1^u(\theta_u, s_u) = W_2^s(\theta_s, s_s) \tag{4.1}$$

In order to solve Eq. (4.1), it would help to be able to restrict our solution search to only certain regions of the $(\theta_u, s_u, \theta_s, s_s)$ space. To this end, we define the concept of layers.

Let λ_u and λ_s be the multipliers for the internal dynamics on W_1^u and W_2^s, respectively. Let $\mathbb{T}\times(-D_u, D_u)$ and $\mathbb{T}\times(-D_s, D_s)$ be the fundamental domains of the parameterizations of W_1^u and W_2^s, respectively. Now, define subsets U_n^+, U_n^- and S_n^+, S_n^- of W_1^u and W_2^s as follows:

$$U_n^+ = \{W_1^u(\theta, s) : (\theta, s) \in \mathbb{T} \times [D_u\lambda_u^{n-1}, D_u\lambda_u^n]\} \tag{4.2}$$

$$U_n^- = \{W_1^u(\theta, s) : (\theta, s) \in \mathbb{T} \times [-D_u\lambda_u^{n-1}, -D_u\lambda_u^n]\} \tag{4.3}$$

$$S_n^+ = \{W_2^s(\theta, s) : (\theta, s) \in \mathbb{T} \times [D_s/\lambda_s^{n-1}, D_s/\lambda_s^n]\} \tag{4.4}$$

$$S_n^- = \{W_2^s(\theta, s) : (\theta, s) \in \mathbb{T} \times [-D_s/\lambda_s^{n-1}, -D_s/\lambda_s^n]\} \tag{4.5}$$

where $n \in \mathbb{Z}$. Finally, define $U_n = U_n^+ \cup U_n^-$ and $S_n = S_n^+ \cup S_n^-$. We refer to the subsets U_n and S_n as layers, and to U_n^+, S_n^+ and U_n^-, S_n^- as positive and negative half-layers, respectively. In our experience, $W_1^u(\theta_u, s_u)$ and $W_2^s(\theta_s, s_s)$ do not intersect for $|s_u| < D_u$ and $|s_s| < D_s$; this can usually be seen from plotting the projections of the manifolds for these s-values in (x, y, p_x) space. Hence, if W_1^u and W_2^s intersect, it must be that U_{n_1} intersects S_{n_2} for some $n_1, n_2 \in \mathbb{Z}^+$.

The most important property of these layers is that $F(U_n) = U_{n+1}$ and $F(S_n) = S_{n-1}$; more generally, $F^k(U_n) = U_{n+k}$ and $F^k(S_n) = S_{n-k}$ for all $k \in \mathbb{Z}$. This allows us to restrict our heteroclinic connection search to only certain pairs of layers of W_1^u and W_2^s. To see this, suppose we are searching for a heteroclinic connection which comes from layer U_{n_1} intersecting layer S_{n_2} at $\mathbf{x} \in \mathbb{R}^4$. Then, since $F(U_n) = U_{n+1}$ and $F(S_n) = S_{n-1}$, we have that $F(\mathbf{x})$ is in the intersection of U_{n_1+1} and S_{n_2-1}. More generally, for all $k \in \mathbb{Z}$, we have that

$$F^k(\mathbf{x}) \in U_{n_1+k} \cap S_{n_2-k} \tag{4.6}$$

Now, if n_1 and n_2 are both odd or both even, using $k = \frac{n_2-n_1}{2}$ in Eq. (4.6) gives us $F^k(\mathbf{x}) \in U_{\tilde{n}} \cap S_{\tilde{n}}$, where $\tilde{n} \overset{\text{def}}{=} \frac{n_1+n_2}{2}$. On the other hand, if n_1 and n_2 are of opposite parity, setting $k = \frac{n_2-n_1+1}{2}$ in Eq. (4.6) gives us $F^k(\mathbf{x}) \in U_{\tilde{n}} \cap S_{\tilde{n}-1}$, where $\tilde{n} \overset{\text{def}}{=} \frac{n_1+n_2+1}{2}$.

When searching for the heteroclinic trajectory which arises due to the manifolds' intersection at $\mathbf{x}$, it is enough to find any point on the orbit of $\mathbf{x}$ under the map F, including $F^k(\mathbf{x})$ from the preceding analysis. Based on the above discussion, it is clear that we will find the point $F^k(\mathbf{x})$ if we look for intersections of pairs of layers of form (U_n, S_n) or (U_n, S_{n-1}) for $n \in \mathbb{Z}^+$ (as mentioned earlier, our experience is that the manifolds do not intersect for $|s_u| < D_u$ and $|s_s| < D_s$, so we only consider positive n). Since $\mathbf{x}$ was an arbitrary heteroclinic point, if we search for intersections of pairs of layers of the form just presented above, we will find all possible heteroclinic trajectories.

As a final note, it is easy to see that if U_{n_1} intersects S_{n_2}, this implies that the time of flight of the resulting heteroclinic connection from the fundamental domain of one torus manifold to the other is $2\pi(n_1+n_2)/\Omega_p$; this is because n_1+n_2 mappings by F are required. Hence, the layer indices can be thought of as a proxy for the connection trajectory time of flight.

4.4 Rapid GPU-Assisted Search for Manifold Intersections

With methods of computing manifolds and restricting the connection search to certain layer pairs now developed, we now seek to develop computationally fast methods of finding intersections of the manifold layers. The manifolds being dealt with are geometric objects in 4D space, so by discretizing the manifolds and applying methods inspired by those from computer graphics collision detection algorithms, we are able to very rapidly search a pair of manifolds for intersections. The algorithms are massively sped up by taking advantage of the huge number of threads available on modern graphics processing units (GPUs). We start this section a with description of the manifold discretization. After this we give the full explanation and demonstration of our intersection search algorithm.

4.4.1 Discrete Mesh Representation of Manifolds

In Chapter 3, we described how it is possible to compute Fourier-Taylor parameterizations $W(\theta, s)$ of the stable and unstable manifolds of stroboscopic map invariant circles. We also gave Equations (3.92) and (3.93) demonstrating how to use the parameterizations to compute $W(\theta, s)$ for s-values outside the fundamental domain $\mathbb{T} \times (-D, D)$. From this, we were able to find a mesh representation of the globalized manifold using the method of Section 3.6.1, which was useful for visualization and plotting. However, this same discrete manifold mesh can also be used for computations and analysis.

The procedure of Section 3.6.1 involved computing the manifold points $W(\theta_i, \lambda^n s_k)$ if $|\lambda| > 1$ or $W(\theta_i, \lambda^{-n} s_k)$ if $|\lambda| < 1$, where $\theta_i = 2\pi i/N$, $i = 0, 1, \ldots, N-1$, the set $\{s_k\}_{k=1}^{L}$ is an evenly spaced grid of L s-values from $-D$ to D, and n ranges from 0 up to some $n_{max} \in \mathbb{Z}^+$. This yields a discretized representation of the manifold subset $\{W(\theta, s) : (\theta, s) \in \mathbb{T} \times [-\sigma, \sigma]\}$, where $\sigma = \lambda^{n_{max}} D$ if $|\lambda| > 1$ and $\sigma = \lambda^{-n_{max}} D$ if $|\lambda| < 1$. Now define the set $\mathcal{S} = \{\lambda^n s_k\}$ (for $|\lambda| > 1$) or $\mathcal{S} = \{\lambda^{-n} s_k\}$ (for $|\lambda| < 1$), $n = 0, 1, \ldots, n_{max}, k = 1, \ldots, L$. The set $\mathcal{S}$ contains all s values for the computed manifold points, and is an unevenly spaced set of real numbers ranging from $s = -\sigma$ to σ. Note that since $s_1 = -D$ and $s_L = D$, the s-values of the U_n layer boundaries ($s = \pm\lambda^n D$) or S_n layer boundaries ($s = \pm\lambda^{-n} D$) will be contained in $\mathcal{S}$; this fact will be useful later.

For later notational convenience, redefine the sequence $\{s_k\}$ now to be $\mathcal{S}$, sorted in ascending order, where now $k = 1, \ldots, M$, $M = |\mathcal{S}|$. With this redefinition, each point computed on the globalized manifold can now be written as $W(\theta_i, s_k)$, indexed using the two integers i between 0 and $N-1$ and k between 1 and M. In our case, we stored the x, y, p_x, and p_y values of the computed manifold points in 4 separate 2D $N \times M$ arrays on

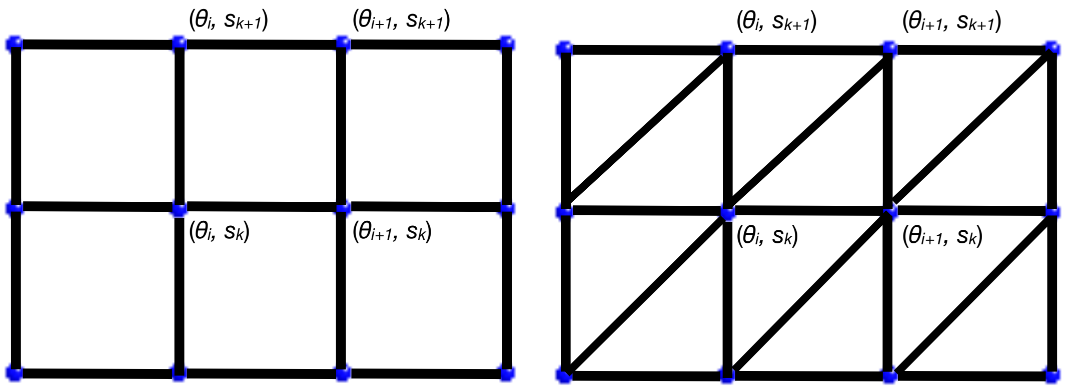


Figure 4.2: Schematic of Quadrilateral (Quad) and Triangular Mesh Construction

the computer, so that the $(i+1,k)$ entry of each array is the x, y, p_x, or p_y coordinate of $W(\theta_i, s_k)$. Moving down a column of each array corresponds to increasing θ and constant s, and moving across a row corresponds to constant θ and increasing s.

With manifold points computed on a discrete grid of (θ, s) ordered pairs and appropriate notation defined, we have everything required in order to define the mesh representation of the manifold. We have the values of $W(\theta, s)$ at the points $\{(\theta_i, s_k)\}$ for $i = 0, \dots, N-1$, $k = 1, \dots, M$. Consider the index i to be modulo N, so that $\theta_N = \theta_0$ and $\theta_{-1} = \theta_{N-1}$. To form the manifold mesh, connect $W(\theta_i, s_k)$ with $W(\theta_{i-1}, s_k)$, $W(\theta_{i+1}, s_k)$, $W(\theta_i, s_{k-1})$, and $W(\theta_i, s_{k+1})$ using line segments. If $k-1$ or $k+1$ is outside the range of allowed indices $1, \dots, M$ (which is true if $k = 1$ or $k = M$, respectively), then omit the corresponding segment from the mesh. This yields a quadrilateral mesh representation of W, as is schematically illustrated on the left of Fig. 4.2. We denote the (i, k) quadrilateral, or quad for short, to be that with vertex set $Q_{ik} = \{W(\theta_i, s_k), W(\theta_{i+1}, s_k), W(\theta_i, s_{k+1}), W(\theta_{i+1}, s_{k+1})\}$, where $i = 0, \dots, N-1$, $k = 1, \dots, M-1$ enumerate the $N(M-1)$ quads in the mesh.

As the vertices of a quad in 4D do not determine a plane, it is necessary to consider each quad as being composed of two triangles for later computational purposes. We split the (i, k) quad into two triangles by connecting the vertex $W(\theta_i, s_k)$ with $W(\theta_{i+1}, s_{k+1})$. Hence, for each ordered pair (i, k), $i = 0, \dots, N-1$, $k = 1, \dots, M-1$, we have two triangles. This gives a triangular mesh for W. The right of Fig. 4.2 shows a schematic

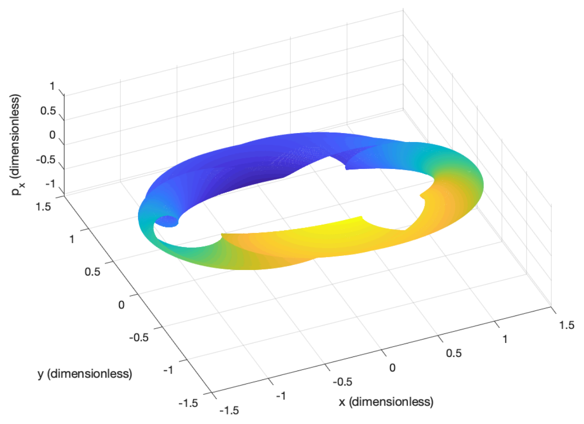


Figure 4.3: (x, y, p_x) projection of 3:4 W^s in Jupiter-Europa PERTBP for $\omega = 1.559620297$

representation of the mesh construction, illustrating all the points which are connected to each other.

A 3D projection of an example globalized stable manifold (denoted W^s) of a 3:4 Jupiter-Europa PERTBP invariant circle is given in Fig. 4.3 (same as Fig. 3.6, repeated here for convenience); the figure was generated using MATLAB's mesh function, which generates a quad mesh similar to the one described here.

4.4.2 GPU-Accelerated Manifold Mesh Intersection Search

Now that we have described how to construct and define the quadrilateral and triangular mesh representations of the manifolds, we start searching for heteroclinic connections. As defined earlier, let $W_1^u(\theta_u, s_u)$ and $W_2^s(\theta_s, s_s)$ represent the unstable and stable manifolds of stroboscopic map invariant circles 1 and 2, respectively. Let λ_u and λ_s be the multipliers for the internal dynamics and $\mathbb{T} \times (-D_u, D_u)$ and $\mathbb{T} \times (-D_s, D_s)$ be the fundamental domains of the parameterizations for W_1^u and W_2^s, respectively. After computing and storing the vertices of the meshes for W_1^u and W_2^s, the problem of finding heteroclinic connections becomes that of finding intersections of these two meshes in 4D space.

From Section 4.3, we know that we can restrict our attention to finding intersections of only certain layers of the manifolds; using the notation from earlier, we seek to find intersections of pairs of layers of the form (U_n, S_n) or (U_n, S_{n-1}). Equivalently, we need to check if U_n^+ or U_n^- intersect any of S_n^+, S_n^-, S_{n-1}^+, or S_{n-1}^-, for $n \in \mathbb{Z}^+$. Recalling that the s-values generated during manifold globalization include those for the boundaries of these layers and half-layers, it turns out that the half-layers just correspond to easily identified subsets of the manifold meshes. For U_n^+, one simply takes the W_1^u mesh vertices which satisfy $s_u \in [D_u\lambda_u^{n-1}, D_u\lambda_u^n]$. For S_n^+, take vertices of the W_2^s mesh with $s_s \in [D_s/\lambda_s^{n-1}, D_s/\lambda_s^n]$. The negative half-layers are the same except for a change in the signs of D_u and D_s. If the manifold coordinates are stored in four 2D arrays as described earlier, with each column containing the coordinates of all points for a given s value, then the vertex set of a half-layer mesh is just comprised of points from a contiguous set of columns.

With the meshes for the half-layers identified, we finally arrive at the problem of searching for intersections of two half-layer meshes. Let us say that the mesh vertices corresponding to an unstable half-layer are $W_1^u(\theta_{u,i}, s_{u,k})$, $i = 0, \ldots, N_1 - 1$, $k = 1, \ldots, M_1$; for the stable half-layer mesh let the vertices be $W_2^s(\theta_{s,j}, s_{s,\ell})$, $j = 0, \ldots, N_2 - 1$, $\ell = 1, \ldots, M_2$. As was done in the previous section, define the quad vertex sets

$$Q_{ik}^u = \{W_1^u(\theta_{u,i}, s_{u,k}), W_1^u(\theta_{u,i+1}, s_{u,k}), W_1^u(\theta_{u,i}, s_{u,k+1}), W_1^u(\theta_{u,i+1}, s_{u,k+1})\} \tag{4.7}$$

$$Q_{j\ell}^s = \{W_2^s(\theta_{s,j}, s_{s,\ell}), W_2^s(\theta_{s,j+1}, s_{s,\ell}), W_2^s(\theta_{s,j}, s_{s,\ell+1}), W_2^s(\theta_{s,j+1}, s_{s,\ell+1})\} \tag{4.8}$$

where $i = 0, \ldots, N_1 - 1$; $j = 0, \ldots, N_2 - 1$; $k = 1, \ldots, M_1 - 1$; and $\ell = 1, \ldots, M_2 - 1$ (again consider the indices i and j to be modulo N_1 and N_2, respectively).

For notational convenience, we also use Q_{ik}^u and $Q_{j\ell}^s$ to refer to the quads formed by the vertices contained therein. As described earlier, we consider each Q_{ik}^u and $Q_{j\ell}^s$ to be comprised of two triangles, defined by vertex sets

$$T_{ik}^{u1} = \{W_1^u(\theta_{u,i}, s_{u,k}), W_1^u(\theta_{u,i+1}, s_{u,k}), W_1^u(\theta_{u,i}, s_{u,k+1})\} \tag{4.9}$$

$$T^{u2}_{ik} = \{W^u_1(\theta_{u,i+1}, s_{u,k}), W^u_1(\theta_{u,i}, s_{u,k+1}), W^u_1(\theta_{u,i+1}, s_{u,k+1})\} \tag{4.10}$$

$$T^{s1}_{j\ell} = \{W^s_2(\theta_{s,j}, s_{s,\ell}), W^s_2(\theta_{s,j+1}, s_{s,\ell}), W^s_2(\theta_{s,j}, s_{s,\ell+1})\} \tag{4.11}$$

$$T^{s2}_{j\ell} = \{W^s_2(\theta_{s,j+1}, s_{s,\ell}), W^s_2(\theta_{s,j}, s_{s,\ell+1}), W^s_2(\theta_{s,j+1}, s_{s,\ell+1})\} \tag{4.12}$$

Again for ease of notation, we also use T^{u1}_{ik}, T^{u2}_{ik}, $T^{s1}_{j\ell}$, and $T^{s2}_{j\ell}$ to refer to the plane triangles formed by the vertices contained therein. We have that $Q^u_{ik} = T^{u1}_{ik} \cup T^{u2}_{ik}$ and $Q^s_{j\ell} = T^{s1}_{j\ell} \cup T^{s2}_{j\ell}$.

Now, the problem of searching for intersections between the half-layers can be solved by checking whether any quad Q^u_{ik} intersects any quad $Q^s_{j\ell}$; we say that Q^u_{ik} intersects $Q^s_{j\ell}$ if any of the triangles T^{u1}_{ik} or T^{u2}_{ik} intersect either of $T^{s1}_{j\ell}$ or $T^{s2}_{j\ell}$. This is just the collision detection problem described in Section 4.2.2 with A as the set of all Q^u_{ik} and B being the set of all $Q^s_{j\ell}$. There are $N_1(M_1 - 1)$ quads in the unstable manifold half-layer mesh, and $N_2(M_2 - 1)$ in the stable manifold half-layer mesh; hence, $N_1N_2(M_1 - 1)(M_2 - 1)$ pairs of quads must be checked for intersection. For an example computation in the PERTBP computing intersections between manifolds of 3:4 and 5:6 resonant invariant circles, we had $N_1 = 1024$, $N_2 = 2048$, and $M_1 = M_2 = 35$, for a total of 2,424,307,712 pairs of quads; each quad pair intersection test involves 4 triangle-triangle checks, which gives 9,697,230,848 pairs of triangles to check for our example. Given the massive number of identical checks to be done, it is clear that a GPU will be well suited to this application.

It is possible to exactly determine whether two 2D triangles intersect in 4D by solving a 4×4 system of linear equations and checking whether the solution satisfies certain conditions, to be described in more detail later (see Section 4.4.2); 4 such tests are required in each quad-quad intersection test. However, solving a different 4×4 system for each of billions of triangle pairs would be extremely computationally expensive, in addition to not being a suitable algorithm to implement on a GPU. Fortunately, as is the expectation in practical collision detection problems, the vast majority of pairs of quads will not intersect when checking two manifold half-layer meshes for intersection. Hence, as described in

Section 4.2.2, it is necessary to first have a computationally cheap broad phase algorithm which can reject most of the non-intersecting quad pairs. For this, we first implement a simple method of grouping quads such that only quads from the same group can possibly intersect; this is then followed by two common broad phase non-intersection tests, both of which can be run on the GPU.

Broad Phase: Uniform Grid Spatial Partitioning

The first step of the broad phase algorithm is aimed at excluding pairs of quads which are in completely different regions of phase space. The main idea [61] is to partition a finite "world" which contains all our objects into a uniform grid of boxes. Then, it is clear that only objects (in our case, quads) overlapping a common box can possibly intersect. Thus, for each box one can make two lists, one of the Q^u_{ik} and another list of the $Q^s_{j\ell}$ which overlap that box; by taking all the pairs of quads having one quad from each list, one gets the set of quad pairs which potentially intersect in that box. Their union over all boxes forms the set of all potentially intersecting quad pairs.

As long as the grid was not too coarse, one will end up with a significantly smaller list of quad pairs after this procedure. This is schematically illustrated in 2D in Figure 4.4 with polygons instead of quads, and $A = B$ (in the framework of Section 4.2.2); here, there are 8 objects, so 28 pairs of objects in the world. The world is partitioned into a uniform grid of 6 boxes, each identified by two grid indices ranging from 1 to 3 in x and 1 to 2 in y. After this, there are only 3 potentially intersecting pairs of objects, one from each of the boxes 12, 13, and 23. Other than cutting the number of quad pairs to check, the key advantage to the spatial partitioning procedure is that finding the lists of quads in each box has complexity $O(|A| + |B|)$ rather than $O(|A||B|)$. To see this, we briefly describe the steps involved.

First of all, we define a finite "world" as a large box containing all the quads, and a uniform grid. The world's minimum and maximum x-bounds can simply be taken as

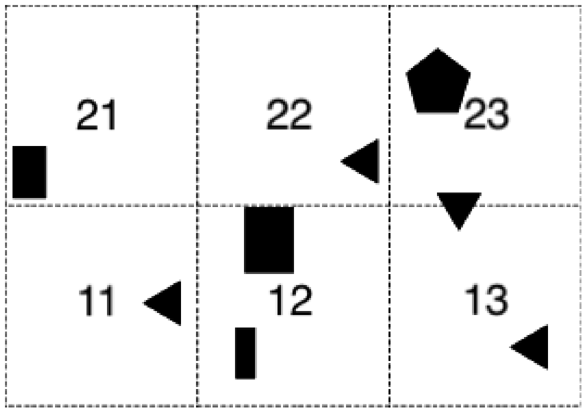


Figure 4.4: Illustration [63] of uniform grid spatial partitioning in 2D space

the minimum x_{min} and maximum x_{max} of x over all the two manifolds' half-layer points $W_1^u(\theta_{u,i}, s_{u,k})$ and $W_2^s(\theta_{s,j}, s_{s,\ell})$. The grid size in x can then be set as $\Delta_x = \frac{x_{max}-x_{min}}{N_x}$ for some $N_x \in \mathbb{Z}^+$; one should choose N_x so that Δ_x is greater than the largest x-width of all quads. y, p_x, and p_y world and grid sizes are done similarly. Then, for each quad, one calculates its overlapped grid indices in each coordinate; for example, given Q_{ik}^u with minimum and maximum x-coordinates $x_{min,ik}$ and $x_{max,ik}$, the overlapped x grid indices are $\lceil \frac{x_{min,ik}-x_{min}}{\Delta_x} \rceil$ and $\lceil \frac{x_{max,ik}-x_{min}}{\Delta_x} \rceil$. The same can be done with the $Q_{j\ell}^s$ as well as with the y, p_x, and p_y grid indices.

Once the overlapped grid indices in x, y, p_x, and p_y have been found for all quads, one forms the lists of Q_{ik}^u and $Q_{j\ell}^s$ overlapping each box. For this, one can simply iterate over all boxes, each of which corresponds to a combination of grid indices, and find all Q_{ik}^u and $Q_{j\ell}^s$ overlapping that same combination of indices. Functions such as MATLAB's find or Julia's findall, which return all true indices of an array, can be useful for this step. It is clear that at no point of this algorithm does one consider pairs of quads; all computations involve only one quad at a time, hence the $O(|A| + |B|)$ complexity rather than $O(|A||B|)$. Finally, from these lists one finds all potentially intersecting quad pairs in each box and in the world as a whole.

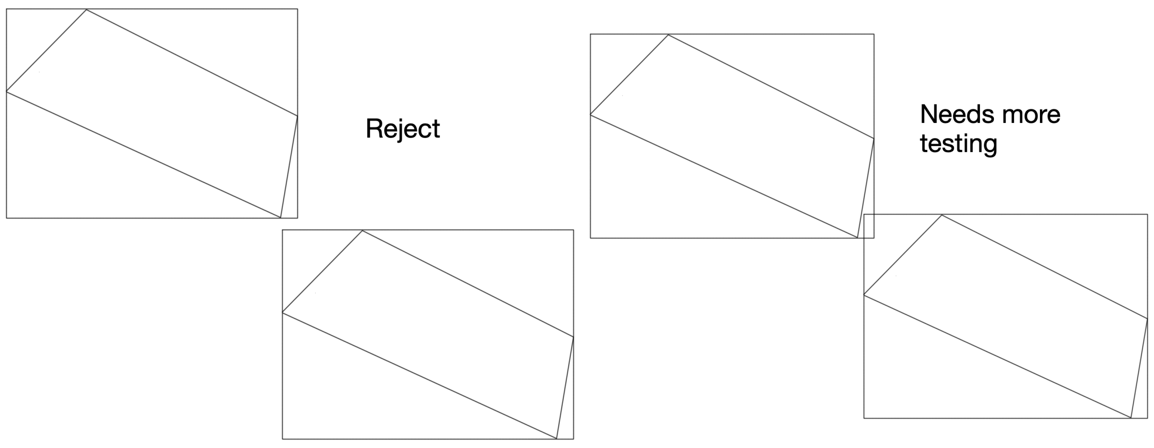


Figure 4.5: Illustration of bounding box test in 2D space

Broad Phase: Bounding Box Test

The next step of the broad phase is a simple axis-aligned bounding box test[61], applied to each of the potentially intersecting pairs of quads from the previous step. This test is generally used with geometric objects in 3D space, but the concept works the same in 4D. The basic idea is to consider each quad to be enclosed by a minimal 4D box (the bounding box) having its edges parallel to the x, y, p_x, and p_y axes. Then, to check whether two quads Q^u_{ik} and $Q^s_{j\ell}$ might intersect, simply check whether their corresponding bounding boxes intersect. If they do not, then we can reject the possibility of the quads intersecting; if the boxes do intersect, then additional testing is required. Figure 4.5 illustrates how this test works in 2D. The test is equivalent to checking whether the maximum x coordinate of the 4 vertices of Q^u_{ik} is less than the minimum x coordinate of the 4 vertices of $Q^s_{j\ell}$; we also reverse the roles of Q^u_{ik} and $Q^s_{j\ell}$ and repeat the check. The same is also done for the y, p_x, and p_y coordinates; if any of the checks are true, then the quads cannot intersect.

Broad Phase: Möller Quick Test

Even after the spatial grid partitioning and bounding box tests have excluded most of the non-intersecting quad pairs, there are still often a fair number of pairs left. Hence, we run a second broad phase test on the remaining pairs which is due to Möller [64]. The full Möller test is a two-part triangle-triangle intersection test developed for applications in 3D graphics; we use only the first part of this test, which is for quick rejection of non-intersecting pairs of triangles. Given two triangles in 3D space, the quick test checks whether all the vertices of one triangle are on the same side of the plane formed by the other; if so, the pair of triangles cannot intersect. In our case, since our objects are in 4D (x, y, p_x, p_y) space, we project the quads and their constituent triangles onto 3D (x, y, p_x) space in order to carry out the test. Of course, if the 3D projected quads and triangles do not intersect, then neither will the full quads and triangles in 4D space.

Denote the projected quads in (x, y, p_x) space as $\tilde{Q}^u_{ik}$ and $\tilde{Q}^s_{j\ell}$, and their constituent projected triangles as $\tilde{T}^{u1}_{ik}$, $\tilde{T}^{u2}_{ik}$ and $\tilde{T}^{s1}_{j\ell}$, $\tilde{T}^{s2}_{j\ell}$. Then, the first step in the Möller quick test is to see whether all four vertices of $\tilde{Q}^s_{j\ell}$ are (1) on the same side of $\tilde{T}^{u1}_{ik}$, and (2) on the same side of $\tilde{T}^{u2}_{ik}$. Both statements (1) and (2) must be true in order to rule out an intersection with $\tilde{Q}^u_{ik}$. To check (1), we first need the equation of the plane in which $\tilde{T}^{u1}_{ik}$ lies. This can be found using standard techniques; the plane will be comprised of all points $\mathbf{X} \in \mathbb{R}^3$ such that

$$f^{u1}_{ik}(\mathbf{X}) \overset{\text{def}}{=} N^{u1}_{ik} \cdot (\mathbf{X} - \tilde{W}^u_1(\theta_{u,i}, s_{u,k})) = 0 \tag{4.13}$$

$$N^{u1}_{ik} = \left[\tilde{W}^u_1(\theta_{u,i+1}, s_{u,k}) - \tilde{W}^u_1(\theta_{u,i}, s_{u,k})\right] \times \left[\tilde{W}^u_1(\theta_{u,i}, s_{u,k+1}) - \tilde{W}^u_1(\theta_{u,i}, s_{u,k})\right]$$

where $\tilde{W}^u_1(\theta_u, s_u)$ denotes the projection of the point $W^u_1(\theta_u, s_u)$ into (x, y, p_x) space. With the equation (4.13) of the plane of $\tilde{T}^{u1}_{ik}$ found, we can determine whether all vertices of $\tilde{Q}^s_{j\ell}$ are on the same side of $\tilde{T}^{u1}_{ik}$ by simply evaluating $f^{u1}_{ik}(\mathbf{X})$ at the four vertices and seeing if the resulting four values all have the same sign. If they do, then (1) is true, otherwise it is not.

To check (2), we do the same procedure, but using the vertices of $\tilde{T}^{u2}_{ik}$ instead of those of $\tilde{T}^{u1}_{ik}$. If both (1) and (2) are true, then we reject the possibility of $Q^s_{j\ell}$ intersecting Q^u_{ik}. Otherwise, we carry out the same test as above, but with the roles of $Q^s_{j\ell}$ and $\tilde{Q}^u_{ik}$ swapped so that we check whether all four vertices of $\tilde{Q}^u_{ik}$ are (1) on the same side of $\tilde{T}^{s1}_{j\ell}$, and (2) on the same side of $\tilde{T}^{s2}_{j\ell}$. If both of these statements are true, then we conclude that $Q^{s}_{j\ell}$ cannot intersect Q^{u}_{ik}; otherwise, we move on to the final, precise test.

Narrow Phase

As described in Section 4.2.2, the broad phase of collision detection is followed by the narrow phase. Thankfully, after the broad phase algorithm is run on the quad pairs generated from two half-layer meshes, the number of potentially intersecting quad pairs left to test is usually quite small. For our earlier example with 2,424,307,712 pairs of quads to be checked for each pair of half-layers, after the broad phase, there would only be at very most a few tens of thousands of cases left to test, and usually far less. In fact, for pairs of half-layers with smaller indices (such as U_2^+, S_2^+), our experience is that the bounding box test alone rejects all of the quad pairs! For those few pairs of quads which have not been rejected in the broad phase, we now need to run a more computationally heavy narrow phase test to check for intersections, as well as computing the intersection if it exists.

Suppose that Q^{u}_{ik} and $Q^{s}_{j\ell}$ are such a pair of quads which passed the broad phase. At this stage, we start dealing exclusively with their constituent triangles; we check whether any of T^{u1}_{ik} or T^{u2}_{ik} intersect either of $T^{s1}_{j\ell}$ or $T^{s2}_{j\ell}$. For this, we need an algorithm to determine whether (and if so, where) two triangles intersect each other in 4D space. Let T_1 and T_2 be two triangles with vertices $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3} \in \mathbb{R}^4$ and $\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3} \in \mathbb{R}^4$, respectively. Then, to determine whether T_1 and T_2 intersect, the first step is to find the intersection of the planes containing these two triangles. The equation to solve to help find this is

$$\mathbf{x_2} + (\mathbf{x_1} - \mathbf{x_2})a + (\mathbf{x_3} - \mathbf{x_2})b = \mathbf{y_2} + (\mathbf{y_1} - \mathbf{y_2})c + (\mathbf{y_3} - \mathbf{y_2})d \tag{4.14}$$

where $a, b, c, d \in \mathbb{R}$ are the quantities to be solved for. Equation (4.14) is a 4D linear equation with 4 unknowns, so it generically admits a unique solution. After solving for $a, b, c,$ and d, it is easy to see that the LHS and RHS of equation (4.14) are themselves equal to the intersection point of the planes containing the two triangles; the values of $a, b, c,$ and d determine whether this intersection point lies on each triangle itself. The conditions for the intersection to be in T_1 and T_2 are simple; we just need $a, b, c, d \geq 0$, $a + b \leq 1$, and $c + d \leq 1$.

If the three conditions just given are not all satisfied, then it is confirmed that T_1 and T_2 do not intersect. If they are all satisfied, then the triangles do intersect and the intersection point is given by either side of Equation (4.14). Furthermore, if T_1 and T_2 are T^{u1}_{ik} or T^{u2}_{ik} and $T^{s1}_{j\ell}$ or $T^{s2}_{j\ell}$, respectively, we can actually use the values of $a, b, c,$ and d to estimate a solution for the equation

$$W_1^u(\theta_u, s_u) = W_2^s(\theta_s, s_s) \tag{4.15}$$

If, for instance, $T_1 = T^{u1}_{ik}$ with $\mathbf{x}_1 = W_1^u(\theta_{u,i+1}, s_{u,k})$, $\mathbf{x}_2 = W_1^u(\theta_{u,i}, s_{u,k})$, $\mathbf{x}_3 = W_1^u(\theta_{u,i}, s_{u,k+1})$; and $T_2 = T^{s1}_{j\ell}$ with $\mathbf{y}_1 = W_2^s(\theta_{s,j+1}, s_{s,\ell})$, $\mathbf{y}_2 = W_2^s(\theta_{s,j}, s_{s,\ell})$, $\mathbf{y}_3 = W_2^s(\theta_{s,j}, s_{s,\ell+1})$, we have

$$(\theta_u, s_u) \approx ((1-a)\theta_{u,i} + a\theta_{u,i+1}, (1-b)s_{u,k} + bs_{u,k+1}) \tag{4.16}$$

$$(\theta_s, s_s) \approx ((1-c)\theta_{s,j} + c\theta_{s,j+1}, (1-d)s_{s,\ell} + ds_{s,\ell+1}) \tag{4.17}$$

We store these approximate solutions of equation (4.15) along with the mesh intersection points.

One thing to note about this search for manifold intersections is that we represent the manifolds using planar meshes, since the mesh faces are triangles. The true manifolds lie close to these triangular faces, but the manifolds are curved rather than planar. Hence, any intersection found from this search will be subject to some error, on the order of the squares of $a, b, c,$ and d. We will describe how to refine these manifold intersections to higher accuracy later in Section 4.5.

4.4.3 Computational Implementation

With the various steps of our mesh intersection algorithm explained, we now describe the implementation of our manifold mesh intersection method in a computer program. Our programs were written using the Julia programming language [65], a relatively new high-level language which has gained significant interest in recent years due to its excellent performance, ease of use, multiple-dispatch features, and variety of high-quality packages (many of which work seamlessly with each other, thanks to multiple dispatch). There is a Julia package called OpenCL.jl [66] which allows one to transfer data to and from an OpenCL device and run OpenCL kernels from Julia programs; the C code for the kernel is simply passed as a large string to an OpenCL.jl function, which generates a kernel which can be run from within Julia. Identification of manifold mesh vertices belonging to a certain half-layer is just a matter of careful indexing; most of the computations occur in trying to detect intersections of two half-layers, so it is this part of the method we focus on here.

As was defined earlier, let $W_1^u(\theta_{u,i}, s_{u,k})$, $i = 0, \ldots, N_1 - 1$, $k = 1, \ldots, M_1$ and $W_2^s(\theta_{s,j}, s_{s,\ell})$, $j = 0, \ldots, N_2 - 1$, $\ell = 1, \ldots, M_2$ be the vertices of the unstable and stable manifold half-layer meshes being considered, respectively. We store the vertex coordinates of each manifold half-layer mesh in four 2D arrays, one for each of x, y, p_x, and p_y; using the convention of 1-based indexing as in MATLAB and Julia, the $(i+1, k)$ entries of the four $N_1 \times M_1$ arrays containing unstable half-layer coordinates are the x, y, p_x, and p_y coordinates of $W_1^u(\theta_{u,i}, s_{u,k})$. Similarly, the $(j+1, \ell)$ entries of the four $N_2 \times M_2$ arrays containing stable half-layer coordinates are the x, y, p_x, and p_y coordinates of $W_2^s(\theta_{u,j}, s_{u,\ell})$. These eight 2D coordinate arrays are provided as inputs to our mesh intersection function, which starts by applying the spatial partitioning scheme of Section 4.4.2 to determine the world, grid, and lists of Q_{ik}^u and $Q_{j\ell}^s$ overlapping each grid box; we store the lists of quads in two arrays of arrays. The m^{th} element of each array of arrays is a 1D array of 0-based linear indices ($i + (k-1)N_1$ for Q_{ik}^u or $j + (\ell - 1)N_2$ for $Q_{j\ell}^s$) identifying the quads overlapping the m^{th} grid box. Note that a box cannot contain a potentially intersecting pair of quads if no Q^u_{ik} or no $Q^s_{j\ell}$ overlaps it; hence, we discard all elements of the arrays of arrays corresponding to such boxes.

The spatial partitioning, since it does not involve pairs and is thus linear in the number of quads, is done on the CPU. However, the remaining broad phase tests apply to the potentially intersecting quad pairs identified by the partitioning; the number of such pairs can still be a fairly large number. Thus the bounding box and quick Möller tests of Sections 4.4.2-4.4.2 are done on the GPU using OpenCL.jl. However, some versions of OpenCL C, including the one for MacOS, do not support arrays of arrays (pointers to pointers). Hence, we first convert the two arrays of 1D arrays of Q^u_{ik}, $Q^s_{j\ell}$ linear indices into two large 1D arrays of quad indices `idxs_quads_1,2` by simply concatenating the quad index arrays over all grid boxes. The `idxs_quads_1,2` arrays are supplemented by auxiliary arrays `grid_box_nums_1,2`, whose m^{th} entries are the number of Q^u_{ik}, $Q^s_{j\ell}$ overlapping the m^{th} box. We also compute the cumulative sum arrays of `grid_box_nums_1,2`, to the results of which we then prepend a 0. The m^{th} entry of the resulting two arrays will be the 0-based starting index in `idxs_quads_1,2` of the m^{th} box's Q^u_{ik}, $Q^s_{j\ell}$ indices list; we thus denote these two arrays by `box_start_idxs_1,2`.

The array `box_start_idxs_1,2` will allow the OpenCL kernel to identify the elements of `idxs_quads_1,2` corresponding to a given box. However, we still need a way of determining which box a given thread should investigate. For this we first compute the element-wise product of `grid_box_nums_1` and `grid_box_nums_2`, which gives an array whose m^{th} entry is the number of potentially intersecting quad pairs in the m^{th} box. Again taking its cumulative sum, we store the end result as an array `box_gid_idxs`; we will be able to use this to assign a box to each OpenCL work item (see lines 3-5 of Algorithm 1). Its last entry gives the total number N_{total} of potentially intersecting quad pairs to be considered. Finally, we transfer the manifold half-layer coordinate arrays as well as `idxs_quads_1,2`, `box_start_idxs_1,2`, and `box_gid_idxs` to the GPU. We also allocate an output buffer `out` of N_{total} 32-bit unsigned integers on the GPU, where we will store identifiers for quad pairs which pass all broad phase tests. We also supply the GPU with a pointer `num_passed` to an integer initialized to zero, which will serve as a counter for how many quad pairs are not rejected.

Once the data is transferred to the GPU and buffers allocated, we execute our bounding box and quick Möller kernel (recall that the narrow phase in this case is unsuitable for GPU implementation). We need a way to determine which two quads Q^u_{ik} and $Q^s_{j\ell}$ each work item should test. Hence, the first few steps of the kernel involve extracting the indices of these quads from `idxs_quads_1,2`. As OpenCL kernels are written in a version of C, the 2D indexing used in Julia does not apply here when reading from the arrays of half-layer coordinates; this is the reason that we stored linear indices in the `idxs_quads_1,2` arrays earlier. Each work item applies the broad phase tests to a different pair of quads. The entire kernel is too long to include in this dissertation, but the essential steps are summarized in Algorithm 1.

Once the broad phase kernel has finished, we read the `num_passed` counter value N_{pass} to find out how many potentially intersecting quad pair identifiers there are in the `out` GPU buffer. We then transfer the first N_{pass} entries from `out` to the host computer memory, after which we convert each pair identifier `pid` back to a pair of linear quad indices by inverting the expression for `out[out_idx]` from line 19 of Algorithm 1; this yields `idx_1` $=$ `pid` $\%\,(N_1 * M_1)$ (for Q^u_{ik}) and `idx_2` $=$ `pid`$/(N_1 * M_1)$ (for $Q^s_{j\ell}$). Finally, using these indices to find the relevant quads' vertex coordinates, we apply the narrow phase test to each potentially intersecting pair of quads listed in `out`. This test is carried out on the CPU, and the resulting intersections are saved as well as the corresponding approximate solutions $(\theta_u, \theta_s, s_u, s_s)$ to Equation (4.15).

4.4.4 Computational Results

The Julia program implementing the previous algorithms was tested on three different consumer-grade machines. Device 1 was a 2017-era laptop with a 2.9GHz quad core Intel

Algorithm 1 Broad phase OpenCL kernel

```
1: int gid ← work item global ID (will range from 0 to $N_{total} - 1$)
2: int bid ← 0
3: while box_gid_idxs[bid] $\leq$ gid do
4:     bid++ (loop determines which box each gid is working with, while making sure enough work items are assigned to each box to process all quad pairs in that box)
5: end while
6: Find starting index in idxs_quads_1,2 for box bid's overlapping $Q^u_{ik}$, $Q^s_{j\ell}$ indices list: box_start_idx_1,2 ← box_start_idxs_1,2[bid]
7: Find number of quads $Q^u_{ik}$, $Q^s_{j\ell}$ overlapping box bid:
   box_num_quads_1,2 ← box_start_idxs_1,2[bid+1]-box_start_idx_1,2
8: Find a linear identifier for which pair of quads in box bid this work item will test:
   pid_box ← gid - box_gid_idxs[bid]
9: Convert the linear identifier pid_box to a pair of quad list indices for box bid. Read the quad lists at those indices to find the corresponding quads' coordinate array linear indices:
   idx_1 ← idxs_quads_1[box_start_idx_1 + (pid_box %box_num_quads_1)]
   idx_2 ← idxs_quads_2[box_start_idx_2 + (pid_box / box_num_quads_1)]
10: Read the coordinates of the $Q^u_{ik}$ and $Q^s_{j\ell}$ vertices from the coordinate arrays using idx_1 and idx_2. Three vertices of $Q^u_{ik}$ will be stored at idx_1, idx_1+1, and idx_1+$N_1$. The fourth will be at idx_1+$N_1$+1 if idx_1%$N_1 \neq -1$, else at idx_1$-N_1$+1. Similar for $Q^s_{j\ell}$.
11: Use fmin/fmax to find smallest/largest $x, y, p_x, p_y$ coordinates of vertices of $Q^u_{ik}$ and $Q^s_{j\ell}$. Carry out bounding box test; store 0 in val if intersection is rejected, otherwise 1.
12: if val == 1 then
13:     Compute normals $N^{u1}_{ik}$ and $N^{u2}_{ik}$. Carry out quick Möller test to check if $\tilde{Q}^s_{j\ell}$ vertices are all on the same side of $\tilde{T}^{u1}_{ik}$ and $\tilde{T}^{u2}_{ik}$ (OpenCL has functions for cross and dot products). If intersection is rejected, val ← 0.
14: end if
15: if val == 1 then
16:     Repeat step 13 with roles of $\tilde{Q}^u_{ik}$ and $\tilde{Q}^s_{j\ell}$ reversed. If intersection is rejected, val ← 0.
17: end if
18: if val == 1 then
19:     Each thread with val true gets a unique consecutive value of out_idx. Store an identifier for the potentially intersecting quad pair $Q^u_{ik}$, $Q^s_{j\ell}$ in the out_idx entry of out.
        out_idx ← atomic_inc(num_passed)
        out[out_idx] ← idx_1 + $N_1$*$M_1$*idx_2
20: end if
```

Table 4.1: Benchmarks for GPU-enabled Manifold Intersection Code (all time values in s)

	Device 1	Device 2	Device 3
Total program runtime	25.93	10.52	12.57
Mean kernel GPU runtime	0.14278	0.02964	0.04160
Kernel time % of total	61.68%	37.07%	31.60%

Table 4.2: Benchmarks for CPU-only Manifold Intersection Code (all time values in s)

	Device 1	Device 2	Device 3
Total program runtime	97.12	60.96	85.90
Mean kernel CPU runtime	0.79743	0.49271	0.70800
Kernel time % of total	91.96%	90.53%	92.31 %

i7-7820HQ CPU and an AMD Radeon Pro 560 GPU with 4GB VRAM. Device 2 was a 2019-era laptop with a 2.6GHz six core Intel i7-9750H CPU and an AMD Radeon Pro 5300M GPU with 4GB VRAM. Device 3 was a desktop tower with a 2011-era 3.33GHz six core Intel Xeon W3680 CPU and a 2016-era Radeon RX 480 GPU with 8GB VRAM. The application used for benchmarking the algorithm was the computation of connections between 3:4 W^u and 5:6 W^s manifolds in the Jupiter-Europa PERTBP, globalized until layers U_{14} and S_{14}. We had $N_1 = 1024$, $N_2 = 2048$, and $M_1 = M_2 = 35$.

We timed the Julia program runtime for this application on all three devices. The program carries out checks up to layer 14, with 8 pairs of half-layers checked per layer, for a total of 112 half-layer pairs checked for intersection during the entire program execution. Each pair of half-layers in turn corresponds to 2,424,307,712 pairs of quads. The resulting program runtimes are given in Table 4.1; already, we can see excellent performance on the 2019-era laptop and the older desktop, with the entire manifold meshes and hundreds of billions of quad pairs being checked for intersection in just over 10 seconds. Even device 1, the older laptop, has reasonable performance as well. As described in Section 4.4.3, the OpenCL kernel is run once for each pair of half-layers being checked for intersections. Thus, the OpenCL kernel is run 112 times throughout the program execution; we timed all of these kernel runs, and give the mean runtimes in Table 4.1 as well.

Although computationally suboptimal, we can force OpenCL.jl to use the CPU for kernel execution instead of the GPU. For the sake of comparison, the results of doing so are given in Table 4.2. From this, we see that the use of the GPU speeds up kernel execution by a factor of 5.6x for device 1, 16.6x for device 2, and 17x for device 3. For the CPU-only program, the kernel executions make up over 90 percent of the overall program runtime; hence, the kernel speedup achieved through GPU usage results in a large speedup of the program as well. The GPU-enabled program is 3.75x faster on device 1, 5.8x faster on device 2, and 6.8x faster on device 3 than the CPU-only program.

As a final note, it is instructive to compare the aforementioned results with those of some of our previous work. The first version [67] of these algorithms for finding manifold intersections was a MATLAB program whose broad phase only consisted of the bounding box test, written as a CPU-only vectorized (and thus parallel) 4D array operation applied to all pairs of quads without any prior grid-based pruning. Running the MATLAB program on device 1, for the same benchmark presented at the beginning of this section, the bounding box test took 8 seconds for each pair of half-layers. The program runtime was thus close to 1000 s.

The second version [15] was a Julia program using OpenCL.jl and GPUs, which did the bounding box and quick Möller tests on the GPU but also did not include the spatial partitioning step in its broad phase. Thus again, the bounding box test was applied to all pairs of quads. In this second study, we had access to the JPL DGX High Performance Computing platform, from which Julia had could use 16 CPU threads and an Nvidia Tesla V100 GPU with 16GB VRAM; this is far more powerful than any of devices 1, 2, or 3. Nevertheless, the same benchmark from earlier in this section took 16 seconds on DGX, which is worse than both devices 2 and 3! This very clearly illustrates the importance of the spatial partitioning for achieving good algorithm performance.

Finally, we show some manifold mesh intersections output by this test application in Figure 4.6; the plot on the left is the zoomed-in projection onto (x, y, p_x) space of the intersections, shown as yellow circles. The plot on the right is the projection onto (x, y, p_y)

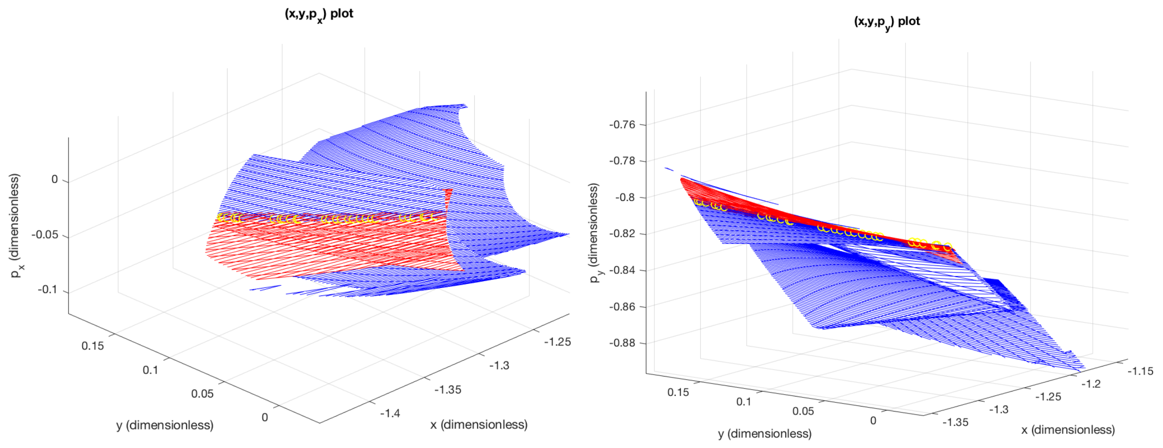


Figure 4.6: 3:4 W^u (red) and 5:6 W^s (blue) heteroclinic connections (yellow circles) of resonant tori in Jupiter-Europa PERTBP

space of the same intersections. These figures are repeated from our previous study [67]; since the benchmark (and the numerical results) in this study are the same as the previous ones, newly generated plots of the mesh intersections found using Julia look exactly the same.

4.5 Refinement of Approximate Manifold Intersections

We have shown how to represent the unstable and stable manifolds W_1^u and W_2^s as meshes, and have also given fast methods for finding intersections of these meshes in 4D space. However, these meshes are made of triangles, which linearly interpolate points between their vertices. Of course, this interpolation has error, so an intersection of the meshes is not an exact heteroclinic connection. We now seek to correct the approximate heteroclinic connections found in the mesh-based search from the previous section. We wish to find solutions $\mathbf{x} = (\theta_u, s_u, \theta_s, s_s)$ of the equation

$$f(\mathbf{x}) = f(\theta_u, s_u, \theta_s, s_s) \overset{\text{def}}{=} W_1^u(\theta_u, s_u) - W_2^s(\theta_s, s_s) = 0 \tag{4.18}$$

As discussed during the description of the precise triangle intersection test, we already will have decent initial guesses for $(\theta_u, s_u, \theta_s, s_s)$ from the mesh search. Hence, we can use differential correction to solve Eq. (4.18), but to do that we must be able to differentiate its LHS.

Denote $\partial_\theta = \frac{\partial}{\partial\theta}$ and $\partial_s = \frac{\partial}{\partial s}$. To differentiate the LHS of Eq. (4.18), we need the partial derivatives $\partial_\theta W_1^u$, $\partial_s W_1^u$, $\partial_\theta W_2^s$, and $\partial_s W_1^u$ evaluated at $(\theta_u, s_u, \theta_s, s_s)$. If $s_u \notin [-D_u, D_u]$ or $s_s \notin [-D_s, D_s]$, we cannot just differentiate the Fourier-Taylor parameterizations of the manifolds and evaluate the result at $(\theta_u, s_u, \theta_s, s_s)$. However, the parameterizations are still of use. Applying Eq. (3.92) to W_1^u and Eq. (3.93) to W_2^s, we have

$$W_1^u(\theta, s) = F^m(W_1^u(\theta - m\omega_1, \lambda_u^{-m} s)) \tag{4.19}$$

$$W_2^s(\theta, s) = F^{-n}(W_2^s(\theta + n\omega_2, \lambda_s^{n} s)) \tag{4.20}$$

where ω_1, ω_2 are the rotation numbers of W_1^u and W_2^s. Differentiating equations (4.19) and (4.20) gives

$$\partial_\theta W_1^u(\theta, s) = DF^m(W_1^u(\theta - m\omega_1, \lambda_u^{-m} s))\, \partial_\theta W_1^u(\theta - m\omega_1, \lambda_u^{-m} s) \tag{4.21}$$

$$\partial_s W_1^u(\theta, s) = \lambda_u^{-m} DF^m(W_1^u(\theta - m\omega_1, \lambda_u^{-m} s))\, \partial_s W_1^u(\theta - m\omega_1, \lambda_u^{-m} s) \tag{4.22}$$

$$\partial_\theta W_2^s(\theta, s) = DF^{-n}(W_2^s(\theta + n\omega_2, \lambda_s^{n} s))\, \partial_\theta W_2^s(\theta + n\omega_2, \lambda_s^{n} s) \tag{4.23}$$

$$\partial_s W_2^s(\theta, s) = \lambda_s^{n} DF^{-n}(W_2^s(\theta + n\omega_2, \lambda_s^{n} s))\, \partial_s W_2^s(\theta + n\omega_2, \lambda_s^{n} s) \tag{4.24}$$

Now, if we choose m and n large enough such that $|\lambda_u^{-m} s_u| < D_u$ and $|\lambda_s^{n} s_s| < D_s$, then one can use equations (4.21)-(4.24) to compute the partials at any $(\theta_u, s_u, \theta_s, s_s)$. Since W_1^u has a Fourier-Taylor series parameterization valid for $|s| < D_u$, we can directly evaluate $W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$. We can also differentiate the parameterization with respect to θ and s to get Fourier-Taylor series for $\partial_\theta W_1^u$ and $\partial_s W_1^u$, which can then both be evaluated at $(\theta, s) = (\theta_u - m\omega_1, \lambda_u^{-m} s_u)$. Finally, the DF^m from equations (4.21) and (4.22) is just a state transition matrix, found by time-$2\pi m/\Omega_p$ numerical integration of the variational equations starting around $W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$. All this allows us to compute the partials of W_1^u; the partials of W_2^s are done very similarly.

We now describe how to compute the various quantities on the RHS of equations (4.21)-(4.24). Without loss of generality, we describe the process for W_1^u; W_2^s is done in the same way. Recall from Eq. (3.73) that our Fourier-Taylor parameterization W_1^u is of the form

$$W_1^u(\theta, s) = \sum_{k \geq 0} W_{1,k}^u(\theta) s^k \tag{4.25}$$

The coefficients $W_{1,k}^u(\theta)$ are stored as arrays of their values at N evenly spaced θ values $\theta_{u,i} = 2\pi i/N$, $i = 0, 1, \ldots, N-1$. Hence, given (θ_u, s_u), we first evaluate $W_1^u(\theta_{u,i}, \lambda_u^{-m} s_u)$ at all the $\theta_{u,i}$ using Eq. (4.25) and the known coefficients $W_{1,k}^u(\theta_{u,i})$. This is followed by a trigonometric interpolation [68] to find the value of $W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$ needed in equations (4.21) and (4.22), which is used to start the numerical integration of the state transition matrix DF^m.

For the RHS of Eq. (4.21), $\partial_\theta W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$ can be found by first using all the $W_1^u(\theta_{u,i}, \lambda_u^{-m} s_u)$ values found earlier to compute $\partial_\theta W_1^u(\theta_{u,i}, \lambda_u^{-m} s_u)$ at all the $\theta_{u,i}$; this can be done from knowledge of $W_1^u(\theta_{u,i}, \lambda_u^{-m} s_u)$ using the general FFT based differentiation technique described by Eq. (3.69) in Section 3.4.10. This is then followed by a trigonometric interpolation to get the value at $\theta = \theta_u - m\omega_1$, which completes the tools required to find $\partial_\theta W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$.

For the RHS of Eq. (4.22), one can find $\partial_s W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$ by first differentiating Eq. (4.25) at each fixed $\theta_{u,i}$ grid value with respect to s. This gives us N polynomials in s

$$\partial_s W_1^u(\theta_{u,i}, s) = \sum_{k \geq 0} (k+1) W_{1,k+1}^u(\theta_{u,i}) s^k \tag{4.26}$$

with known coefficients. Next, the $\partial_s W_1^u(\theta_{u,i}, s)$ series can be evaluated at $s = \lambda_u^{-m} s_u$ for all the $\theta_{u,i}$, finally followed by trigonometric interpolation to find $\partial_s W_1^u(\theta_u - m\omega_1, \lambda_u^{-m} s_u)$.

With all quantities from the RHS of equations (4.21) and (4.22) found, we can now compute the desired partials at (θ_u, s_u). As mentioned earlier, the partials of W_2^s can be found in the exact same manner, after which we can solve Eq. (4.18). With $\mathbf{x} = (\theta_u, s_u, \theta_s, s_s)$ and letting $\mathbf{x}_0$ be the initial guess found earlier for $\mathbf{x}$ solving Eq. (4.18), we use the damped Newton method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha Df^{-1}(\mathbf{x}_k)f(\mathbf{x}_k) \tag{4.27}$$

to differentially correct $\mathbf{x}$ until we have a solution to Eq. (4.18) within tolerance. Here, $Df = [\partial_{\theta_u} f \ \partial_{s_u} f \ \partial_{\theta_s} f \ \partial_{s_s} f]$, and $0 < \alpha < 1$. Trial and error is used to find a value of α such that the iteration converges. We have used α values anywhere from 0.5 to 0.01 in our computations.

Using the damped Newton method, we have been able to differentially correct several of the approximate intersections $(\theta_u, s_u, \theta_s, s_s)$ found in the mesh search benchmark described in the previous section, from an error of 0.01 in Eq. (4.18) to errors of less than 10^{-7}. An example differential correction is displayed in Fig. 4.7, for one of the connections shown in Fig. 4.6. The initial guess in Fig. 4.7 is shown in green, the iterates in yellow, and the final converged solution in cyan. We changed the value of α at one point during the iteration, hence the uneven spacing of the iterates. Also, note that the damped Newton iterates move a large distance away from the initial guess found in the mesh search. This is not unexpected; the derivative of the LHS of Eq. (4.18) is almost singular, since PCRTBP manifold intersections (and hence solutions to Eq. (4.18) in the PCRTBP case) occur along 1D curves rather than at isolated points. In the PCRTBP case, this implies that the derivative at a solution of the equation would actually be singular. The perturbation in the Jupiter-Europa PERTBP is quite weak, so near-singularity is reasonable to expect. It is because of this that the damped Newton method is necessary.

In addition, we found that all of the manifold intersections shown in Fig. 4.6, upon differential correction, actually converged to the same refined solution (the green circle in Fig. 4.7)! This is despite all of the solutions from Fig. 4.6 satisfying Eq. (4.18) with an error of 0.01 or less. Hence, we see that intersecting the discrete mesh representations

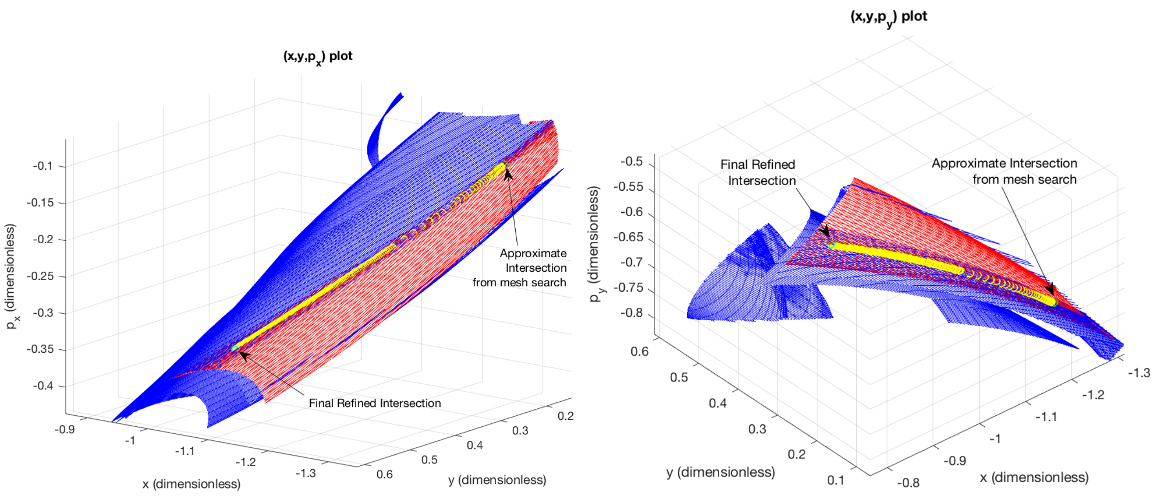


Figure 4.7: 3D projections of iterates of damped Newton method for refinement of an approximate connection found from mesh search

of the manifolds may actually find mostly intersections which correspond to near-misses rather than true intersections of the manifolds, especially when the periodic perturbation is weak. This, as well as the distance between the initial guess and the refined solution in Fig. 4.7, demonstrates the importance of carrying out the differential correction in order to find the true intersections. The mesh-based search is necessary in order to quickly narrow down potential points of interest, but the few final accurate intersections must be found by solving Eq. (4.18) directly using methods such as those described here.

4.6 Conclusions

In this chapter, we presented a suite of concepts, methods, and tools for finding heteroclinic connections between unstable invariant tori in periodically-perturbed PCRTBP models. Using the idea of layers, we can restrict the connection search to only certain subsets of the manifolds. By generating a discrete mesh of points from the manifolds during globalization, one can bring to bear the massively parallel computing power of modern GPUs for the purpose of rapidly detecting and computing intersections of the meshes. Finally, we showed how to refine the solutions found from the mesh search for greater accuracy, using the manifold parameterizations to enable the application of a differential correction algorithm.

Testing our GPU assisted mesh intersection tools, we saw speedups by a factor of over 30 as compared to CPU-only tools. Using an HPC system with a more powerful GPU allows for the checking of 14 layers of two manifolds for intersection in just a matter of seconds. These tools are suitable for exploring many different periodically-perturbed PCRTBP models, as well as the spatial CRTBP, which are areas of ongoing and future work.

Appendices

APPENDIX A
PROOF OF VANISHING E_{CC} AVERAGE

In Section 3.4.5, it was mentioned that the average of $E_{CC}(\theta)$ goes to zero with each quasi-Newton step. We can prove this using a method somewhat inspired by the vanishing lemma proof of [43]. For ease of notation, denote this average as $\lambda_c = \hat{E}_{CC}(0)$, and $\tilde{E}_{CC}(\theta) = E_{CC}(\theta) - \lambda_c$, so that $\tilde{E}_{CC}$ has zero average. Also write $\mathbf{e}_2 = [0\ 1\ 0\ 0]^T$.

Proof. Let $\mathbf{v}_c(\theta)$ denote the second column of P, and define $E_C(\theta) = \left[E_{LC}\ \tilde{E}_{CC}\ E_{SC}\ E_{UC}\right]^T$; note that $E_C(\theta) + \lambda_c\mathbf{e}_2$ is simply the second column of $E_{red}(\theta)$. Left multiplying the definition of E_{red} (Eq. (3.13)) by $P(\theta+\omega)$ and taking column 2 of the result gives

$$P(\theta+\omega)\left(E_C(\theta) + \lambda_c\mathbf{e}_2\right) = DF(K(\theta))\mathbf{v}_c(\theta) - \mathbf{v}_c(\theta+\omega) - T(\theta)DK(\theta+\omega) \quad \text{(A.1)}$$

Define $\mathcal{E}_C(\theta) = P(\theta+\omega)E_C(\theta)$, and note that $P(\theta+\omega)\mathbf{e}_2 = \mathbf{v}_c(\theta+\omega)$. Thus, Eq. (A.1) gives

$$DF(K(\theta))\mathbf{v}_c(\theta) = (1+\lambda_c)\mathbf{v}_c(\theta+\omega) + T(\theta)DK(\theta+\omega) + \mathcal{E}_C(\theta) \quad \text{(A.2)}$$

Now, differentiating Eq. (3.12) yields $DF(K(\theta))DK(\theta) = DK(\theta+\omega) + DE(\theta)$. As mentioned in the proof of Lemma 4, F satisfies $\Omega(\mathbf{v}_1, \mathbf{v}_2) = \Omega(DF(K(\theta))\mathbf{v}_1, DF(K(\theta))\mathbf{v}_2)$ for all $\mathbf{v}_1$, $\mathbf{v}_2 \in \mathbb{R}^4$, where Ω is the symplectic form defined by $\Omega(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T J\mathbf{v}_2$. Thus,

$$\begin{aligned}\Omega(\mathbf{v}_c(\theta), DK(\theta)) &= \Omega(DF(K(\theta))\mathbf{v}_c(\theta), DF(K(\theta))DK(\theta)) \\ &= \Omega\Big((1+\lambda_c)\mathbf{v}_c(\theta+\omega) + T(\theta)DK(\theta+\omega) + \mathcal{E}_C(\theta), DK(\theta+\omega) + DE(\theta)\Big) \\ &= (1+\lambda_c)\Omega\Big(\mathbf{v}_c(\theta+\omega), DK(\theta+\omega)\Big) + \mathcal{O}(DE(\theta)) + \mathcal{O}(\mathcal{E}_C(\theta))\end{aligned} \quad \text{(A.3)}$$

where we use $\Omega(DK(\theta+\omega), DK(\theta+\omega)) = 0$ to get the last line. This yields

$$\int_0^{2\pi} \Omega(\mathbf{v}_c(\theta), DK(\theta))\, d\theta = (1+\lambda_c) \int_0^{2\pi} \Omega\big(\mathbf{v}_c(\theta+\omega), DK(\theta+\omega)\big)\, d\theta \\ + \mathcal{O}(DE(\theta)) + \mathcal{O}(\mathcal{E}_C(\theta)) \tag{A.4}$$

Recognizing that $\int_0^{2\pi} \Omega(\mathbf{v}_c(\theta), DK(\theta))\, d\theta = \int_0^{2\pi} \Omega(\mathbf{v}_c(\theta+\omega), DK(\theta+\omega))\, d\theta$, we have

$$\lambda_c \int_0^{2\pi} \Omega(\mathbf{v}_c(\theta), DK(\theta))\, d\theta = \mathcal{O}(DE(\theta)) + \mathcal{O}(\mathcal{E}_C(\theta)) \tag{A.5}$$

Now, for E and E_{red} small enough, $\mathbf{v}_c(\theta)$ is an approximate symplectic conjugate to $DK(\theta)$. This means that $\Omega(\mathbf{v}_c(\theta), DK(\theta)) \approx 1$ (see Eq. (3.63)), so $\int_0^{2\pi} \Omega(\mathbf{v}_c(\theta), DK(\theta))\, d\theta = \mathcal{O}(1)$. Hence, it must be that $\lambda_c = \mathcal{O}(DE(\theta)) + \mathcal{O}(\mathcal{E}_C(\theta))$, so that as the quasi-Newton method reduces $DE(\theta)$ and $E_C(\theta)$ (and thus also $\mathcal{E}_C(\theta)$) to zero, λ_c goes to zero as well. □

When carrying out the quasi-Newton step of Section 3.4.5 for correcting P and Λ, Eqs. (3.32), (3.38), and (3.41) can be solved exactly (including for non-zero averages on the LHS), which quadratically reduces the E_{LC}, E_{SC}, and E_{UC} components of $E_C(\theta)$ (using the definitions given in the above proof). On the other hand, Eq. (3.35) for E_{CC} can be written as

$$-E_{CC}(\theta) = -\tilde{E}_{CC}(\theta) - \lambda_c = Q_{CC}(\theta) - Q_{CC}(\theta+\omega) \tag{A.6}$$

As mentioned near the end of Section 3.4.5, we ignore the nonzero LHS average $-\lambda_c = -\hat{E}_{CC}(0)$ when solving for Q_{CC}. Thus, what happens is that the zero-average part $\tilde{E}_{CC}(\theta)$ is quadratically reduced by the quasi-Newton step, but λ_c may initially remain in E_{red}. However, the quadratic reductions in E_{LC}, $\tilde{E}_{CC}(\theta)$, E_{SC}, and E_{UC}, and subsequently also in $E(\theta)$ during the following K-correction step, quadratically reduce $E_C(\theta)$ and $DE(\theta)$. This necessitates a reduction in λ_c as described at the end of the above proof.

REFERENCES

[1] H. Poincaré, *Les méthodes nouvelles de la mécanique céleste*. Gauthier-Villars, 1892, vol. 3.

[2] R. R. Bate, D. D. Mueller, and J. E. White, *Fundamentals of Astrodynamics*. New York: Dover Publications, 1971.

[3] M. W. Lo *et al.*, "Genesis mission design," *The Journal of the Astronautical Sciences*, vol. 49, no. 1, pp. 169–184, 2001.

[4] D. Shepelyansky, "Chirikov criterion," *Scholarpedia*, vol. 4, no. 9, p. 8567, 2009, revision #152383.

[5] A. Celletti, *Stability and Chaos in Celestial Mechanics* (Astronomy and Planetary Sciences). Springer-Verlag Berlin Heidelberg, Jan. 2010.

[6] *The circular-restricted three-body problem (CRTBP)*, from http://ccar.colorado.edu/geryon/CRTBP.html, Retrieved December 7 2018.

[7] C. Simó, G. Gómez, À. Jorba, and J. Masdemont, "The bicircular model near the triangular libration points of the RTBP," in *From Newton to chaos*, Springer, 1995, pp. 343–370.

[8] M. Andreu, "The quasi-bicircular problem," Ph.D. dissertation, Citeseer, 1998.

[9] D. Scheeres, "The restricted Hill four-body problem with applications to the Earth–Moon–Sun system," *Celestial Mechanics and Dynamical Astronomy*, vol. 70, no. 2, pp. 75–98, 1998.

[10] L. Hiday-Johnston and K. Howell, "Transfers between libration-point orbits in the elliptic restricted problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 58, no. 4, pp. 317–337, 1994.

[11] V. Szebehely, *Theory of Orbits: The Restricted Problem of Three Bodies.* Academic Press Inc., New York, 1967, p. 668.

[12] À. Haro, M. Canadell, J. Figueras, A. Luque, and J. Mondelo, *The Parameterization Method for Invariant Manifolds: From Rigorous Results to Effective Computations* (Applied Mathematical Sciences). Springer International Publishing, 2016, vol. 195, ISBN: 9783319296623.

[13] B. Kumar, R. L. Anderson, and R. de la Llave, "High-order resonant orbit manifold expansions for mission design in the planar circular restricted 3-body prob-

lem," *Communications in Nonlinear Science and Numerical Simulation*, vol. 97, p. 105 691, 2021.

[14] B. Kumar, R. L. Anderson, and R. de la Llave, "Rapid and accurate methods for computing whiskered tori and their manifolds in periodically perturbed planar circular restricted 3-body problems," *Celestial Mechanics and Dynamical Astronomy*, vol. 134, no. 1, p. 3, 2022.

[15] B. Kumar, R. L. Anderson, and R. de la Llave, "Using GPUs and the parameterization method for rapid search and refinement of connections between tori in periodically perturbed planar circular restricted 3-body problems," in *AAS/AIAA Space Flight Mechanics Meeting*, Feb. 2021.

[16] R. L. Anderson and M. W. Lo, "Dynamical systems analysis of planetary flybys and approach: Planar Europa orbiter," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 6, pp. 1899–1912, 2010. eprint: https://doi.org/10.2514/1.45060.

[17] R. Anderson and M. Lo, "A dynamical systems analysis of resonant flybys: Ballistic case," *The Journal of the Astronautical Sciences*, vol. 58, Apr. 2011.

[18] M. Vaquero and K. C. Howell, "Leveraging resonant-orbit manifolds to design transfers between libration-point orbits," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1143–1157, 2014. eprint: https://doi.org/10.2514/1.62230.

[19] M. Vaquero, Y. Hahn, P. Stumpf, P. N. Valerino, S. V. Wagner, and M. Wong, "Cassini maneuver experience for the fourth year of the solstice mission," in *AIAA/AAS Astrodynamics Specialist Conference*. 2014. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2014-4348.

[20] R. L. Anderson, S. Campagnola, D. Koh, T. P. McElrath, and R. M. Woollands, "Endgame design for Europa lander: Ganymede to Europa approach," *The Journal of the Astronautical Sciences*, vol. 68, no. 1, pp. 96–119, 2021.

[21] R. L. Anderson, S. Campagnola, and G. Lantoine, "Broad search for unstable resonant orbits in the planar circular restricted three-body problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 124, no. 2, pp. 177–199, Feb. 2016.

[22] J. Guckenheimer and P. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Applied Mathematical Sciences). Springer, New York, NY, 1983, vol. 42, ISBN: 978-1-4612-7020-1.

[23] X. Cabré, E. Fontich, and R. de la Llave, "The parameterization method for invariant manifolds III: Overview and applications," *Journal of Differential Equations*, vol. 218, no. 2, pp. 444–515, 2005.

[24] M. Brown and W. D. Neumann, "Proof of the poincaré-birkhoff fixed point theorem.," *Michigan Math. J.*, vol. 24, no. 1, pp. 21–31, 1977.

[25] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, "Heteroclinic connections between periodic orbits and resonance transitions in celestial mechanics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 10, no. 2, pp. 427–469, 2000. eprint: https://doi.org/10.1063/1.166509.

[26] D. Pérez-Palau, J. J. Masdemont, and G. Gómez, "Tools to detect structures in dynamical systems using jet transport," *Celestial Mechanics and Dynamical Astronomy*, vol. 123, no. 3, pp. 239–262, Nov. 2015.

[27] *GNU Scientific Library*, from https://www.gnu.org/software/gsl/doc/html/, Retrieved October 2 2019.

[28] D. Pérez Palau, "Dynamical transport mechanisms in celestial mechanics and astrodynamics problems," Ph.D. dissertation, Universitat de Barcelona, 2015.

[29] T. Bayer, B. Cooke, I. Gontijo, and K. Kirby, "Europa clipper mission: The habitability of an icy moon," in *2015 IEEE Aerospace Conference*, 2015, pp. 1–12.

[30] O. Grasset *et al.*, "Jupiter icy moons explorer (juice): An esa mission to orbit ganymede and to characterise the jupiter system," *Planetary and Space Science*, vol. 78, pp. 1–21, 2013.

[31] J. D. Mireles James and M. Murray, "Chebyshev-Taylor parameterization of stable/unstable manifolds for periodic orbits: Implementation and applications," *International Journal of Bifurcation and Chaos*, vol. 27, no. 14, p. 1 730 050, 2017. eprint: https://doi.org/10.1142/S0218127417300506.

[32] A. Farrés, À. Jorba, and J.-M. Mondelo, "Numerical study of the geometry of the phase space of the augmented Hill three-body problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 129, no. 1, pp. 25–55, 2017.

[33] Z. P. Olikara, "Computation of quasi-periodic tori and heteroclinic connections in astrodynamics using collocation techniques," Ph.D. dissertation, University of Colorado Boulder, 2016.

[34] G. Gómez, J. Llibre, R. Martínez, and C. Simó, *Dynamics and Mission Design Near Libration Points*, 4 vols. Roma: World Scientific, 2001.

[35] N. Bosanac, "Bounded motions near resonant orbits in the Earth-Moon and Sun-Earth systems," in *AAS/AIAA Astrodynamics Specialist Conference, Snowbird, Utah*, 2018.

[36] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, *Dynamical systems, the three-body problem and space mission design*. Marsden Books, 2011.

[37] G. Huguet, R. de la Llave, and Y. Sire, "Computation of whiskered invariant tori and their associated manifolds: New fast algorithms," *Discrete & Continuous Dynamical Systems - A*, vol. 32, no. 4, pp. 1309–1353, 2012.

[38] N. Fenichel, "Persistence and smoothness of invariant manifolds for flows," *Indiana University Mathematics Journal*, vol. 21, no. 3, pp. 193–226, 1971.

[39] M. J. Capiński, M. Gidea, and R. de la Llave, "Arnold diffusion in the planar elliptic restricted three-body problem: Mechanism and numerical verification," *Nonlinearity*, vol. 30, no. 1, p. 329, 2016.

[40] L. Zhang and R. de la Llave, "Transition state theory with quasi-periodic forcing," *Communications in Nonlinear Science and Numerical Simulations*, vol. 62, pp. 229–243, Sep. 2018.

[41] A. Haro and J. Mondelo, "Flow map parameterization methods for invariant tori in hamiltonian systems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 101, p. 105 859, 2021.

[42] M. W. Hirsch, C. C. Pugh, and M. Shub, *Invariant manifolds* (Lecture Notes in Mathematics). Berlin: Springer, 1977.

[43] E. Fontich, R. de la Llave, and Y. Sire, "Construction of invariant whiskered tori by a parameterization method. part i: Maps and flows in finite dimensions," *Journal of Differential Equations*, vol. 246, no. 8, pp. 3136–3213, 2009.

[44] R. de la Llave, A. González, À. Jorba, and J. Villanueva, "KAM theory without action-angle variables," *Nonlinearity*, vol. 18, no. 2, pp. 855–895, Jan. 2005.

[45] R. De la Llave, "A tutorial on KAM theory," in *Smooth Ergodic Theory and Its Applications, Seattle, WA, 1999*, American Mathematical Society, vol. 69, Providence, RI, 2001, pp. 175–292, ISBN: 9780821826829.

[46] C. Chicone, *Ordinary differential equations with applications*. Springer Science & Business Media, 2006, vol. 34.

[47] H. Rüssmann, "On optimal estimates for the solutions of linear partial differential equations of first order with constant coefficients on the torus," in *Dynamical Systems, Theory and Applications: Battelle Seattle 1974 Rencontres*, J. Moser, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 598–624, ISBN: 978-3-540-37505-0.

[48] C. Golé, *Symplectic Twist Maps*. World Scientific, 2001. eprint: https://www.worldscientific.com/doi/pdf/10.1142/1349.

[49] L. V. Ahlfors, *Complex analysis : an introduction to the theory of analytic functions of one complex variable* (International series in pure and applied mathematics), 3rd. McGraw-Hill, New York, 1979.

[50] A. Haro and R. de la Llave, "A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: Explorations and mechanisms for the breakdown of hyperbolicity," *SIAM J. Appl. Dyn. Syst.*, vol. 6, no. 1, pp. 142–207, 2007.

[51] W. Thirring, *A course in mathematical physics. Walter Thirring ; translated by Evans M. Harrell [t.] 1 and 2, Classical dynamical systems and classical field theory*, 2nd ed. New York; Springer, 1992, ISBN: 0387976094.

[52] À. Haro and R. de la Llave, "A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: Numerical algorithms," *Discrete & Continuous Dynamical Systems - B*, vol. 6, no. 6, pp. 1261–1300, 2006.

[53] M. Rasotto *et al.*, "Differential algebra space toolbox for nonlinear uncertainty propagation in space dynamics," in *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, Mar. 2016.

[54] M. Berz and K. Makino, "Verified integration of ODEs and flows using differential algebraic methods on high-order taylor models," *Reliable Computing*, vol. 4, no. 4, pp. 361–369, 1998.

[55] M. Galassi *et al.*, *GNU Scientific Library Reference Manual - Third Edition*, 3rd. Network Theory Ltd., 2009, ISBN: 0954612078.

[56] A. Celletti, "Basics of regularization theory," in *Chaotic Worlds: From Order to Disorder in Gravitational N-Body Dynamical Systems*, B. A. Steves, A. J. Maciejewski, and M. Hendry, Eds., Dordrecht: Springer Netherlands, 2006, pp. 203–230, ISBN: 978-1-4020-4706-0.

[57] C. Rackauckas and Q. Nie, "Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia," *The Journal of Open Research Software*, vol. 5, no. 1, 2017, Exported from https://app.dimensions.ai on 2019/05/05.

[58] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.

[59] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for?” *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.

[60] Khronos OpenCL Working Group, *The OpenCL specification, version 1.2*, ed. by A. Munshi, 2012.

[61] C. Ericson, *Real-time collision detection* (The Morgan Kaufmann Series in Interactive 3D Technology). Amsterdam: Elsevier : Morgan Kaufmann, 2005, ISBN: 1558607323 9781558607323.

[62] S. Le Grand, “Broad-phase collision detection with cuda,” in *GPU Gems 3*, H. Nguyen, Ed., Addison-Wesley Professional, 2007, ch. 32, ISBN: 9780321545428.

[63] M. Figueiredo, L. Marcelino, and T. Fernando, “A survey on collision detection techniques for virtual environments,” *Proc. of V Symposium in Virtual Reality, Brasil*, vol. 307, Jan. 2002.

[64] T. Möller, “A fast triangle-triangle intersection test,” *Journal of Graphics Tools*, vol. 2, no. 2, pp. 25–30, 1997. eprint: https://doi.org/10.1080/10867651.1997.10487472.

[65] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[66] *OpenCL.jl*, from https://github.com/JuliaGPU/OpenCL.jl, Retrieved Jan 16 2020.

[67] B. Kumar, R. L. Anderson, and R. de la Llave, “Rapid and accurate computation of invariant tori, manifolds, and connections near mean motion resonances in periodically perturbed planar circular RTBP models,” in *AAS/AIAA Astrodynamics Specialist Conference*, 2020.

[68] *triginterp.m*, from http://www.math.udel.edu/~braun/M428/Matlab/interpolation/triginterp.m, Retrieved Jan 16 2020.

VITA

Bhanu Kumar completed his PhD from the School of Mathematics at Georgia Tech, advised by Prof. Rafael de la Llave. He was a recipient of a NASA Space Technology Research Fellowship (NSTRF) for his graduate studies, and also has M.S. and B.S. degrees in Aerospace Engineering from the Daniel Guggenheim School of Aerospace Engineering at Georgia Tech. He was an NSTRF visiting technologist in the Mission Design and Navigation section at the NASA Jet Propulsion Laboratory (JPL), California Institute of Technology, where his mentor and research collaborator was Dr. Rodney Anderson. Prior to joining the Mission Design and Navigation section, he completed yearly co-op work terms in the Radio Science Systems group, also at JPL, between 2014 and 2017. Starting Fall 2022, he will be an NSF postdoctoral research fellow at JPL.