COMPUTATION AND VISUALIZATION OF PERIODIC ORBITS IN THE CIRCULAR RESTRICTED THREE-BODY PROBLEM

CHENGHAI ZHANG

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 9, 2004

© CHENGHAI ZHANG, 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-94757-2
Our file Notre référence
ISBN: 0-612-94757-2

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Computation and Visualization of Periodic Orbits in the Circular Restricted Three-Body Problem

Chenghai Zhang

In this thesis, the continuation and bifurcation software AUTO is used to compute periodic solutions of the circular restricted Three-Body problem (CR3BP). Periodic solution families for the Sun-Earth, the Earth-Moon, and the Sun-Jupiter system are studied in detail. Bifurcation diagrams for these systems are presented. Corresponding periodic orbits are also shown.

To understand the solution structure better, a new data visualization package, PLAUT04, has been developed for AUTO. It reads AUTO data files and creates solution diagrams and bifurcation diagrams. This new package can also be used to animate solutions. A special version of PLAUT04, called PLAUT04/r3b, has been developed for the CR3BP. Using PLAUT04/r3b, we can animate solutions both in a rotating frame and in an inertial frame. These new graphics packages for AUTO have good rendering speed, flexibility, and display quality. A user-friendly interface makes both easy to learn and use.

Acknowledgments

I would like to express my sincerest appreciation to my supervisor professor Eusebius J. Doedel for his support, guidance, patience, and valuable insight, which have made the completion of my thesis possible.

Thanks to Alexei Aganin of Florida State University for his Open Inventor demo.

Thanks to D. J. Dichmann for his email on converting CR3BP data from the rotating frame to the inertial frame.

Thanks to Volodymyr Romanov for providing the code for the calculation of the libration points in the CR3BP.

A special thanks to my wife, Xiaoying Jiang. Her encouragement and love sustains me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, a brief introduction to the circular restricted Three-Body problem (CR3BP) is given. The history and the current status of CR3BP research, and the objectives of this thesis are also discussed.

1.1 The N-Body problem

The CR3BP is a special case of the famous N-Body problem, one of the oldest and most challenging problems in the history of mathematics. Its study requires knowledge of dynamical systems, geometry, physics and analysis. The classical *N-Body problem* states that given the initial positions and velocities of n objects that attract one another by their mutual gravitational forces only, one has to determine their positions and velocities at any time in the future. Thus, if we can solve the N-Body problem, then we can calculate the positions of the objects at a later time. This is useful in Celestial Mechanics, and has applications to space exploration.

The N-Body problem has been actively studied for centuries. Whenever there are several bodies, *e.g.*, planets, stars, or electrons, that move under the force of physical laws, then we have an example of the N-Body problem. When the ancient Greeks studied the movement of the planets, or how gravity works here on the Earth, they were actually trying to understand the N-Body problem.

Newton was the first person to formulate the N-Body problem precisely. Although many famous scientists, such as Newton, Copernicus, Kepler, Lagrange, Poincaré, contributed to the N-Body problem, there remain many unresolved issues.

1.2 The One-Body and the Two-Body problems

The simplest case of the N-Body problem is the *One-Body problem*, *i.e.*, there is only one body which has an initial position and an initial velocity. If an external force is applied to it, the motion of the body changes. The solution of the One-Body problem is easy to obtain from basic physics. *Newton's first law of motion* tells us that an object at rest remains at rest, and an object in motion remains in motion, unless acted upon by an outside force. Newton's great discovery has been of fundamental importance in Science and Engineering.

Next, consider the gravitational interaction between two isolated objects, which do not interact with anything else. For example, let the two bodies be the Earth and the Sun. We know that the Earth orbits around the Sun in an elliptical orbit, while the Sun remains almost idle at the focus of the orbit. This is an example of the *Two-Body problem.* The Two-Body problem is also simple enough to be solved, but it requires more knowledge of calculus and analytic geometry. Newton also solved the Two-Body problem in the form of *Newton's law of gravitation.* People have applied Newton's laws to planets, comets, asteroids, *etc.*

1.3 The Three-Body problem

Based on the previous Earth-Sun example, if we add the Moon to our system as the third body to build a Three-Body system, how do they affect each other? Because each body in this Three-Body system is attracted by the two other bodies, the motion of the bodies becomes much more complicated, and it is much more difficult to predict the positions of the bodies at a later time. Many questions remain unanswered for the *Three-Body problem.*

To make the problem easier, scientists have made simplifying assumptions. These lead to many subproblems of the Three-Body problem. One of these is the CR3BP, which we focus on in this thesis. A detailed discussion of the CR3BP is given in Chapter 2.

1.4 A brief history

Because of the complexity of the Three-Body problem, mathematicians realized that they had to simplify the problem by looking at special cases, for example, the case where one particle is much less massive than the other two, or the case where two particles are much less massive than the third.

Pierre-Simon de Laplace solved a special case, but failed at the general one. Henri Poincaré tried his hands at the problem, but failed in his attempt too.

Weierstrass posed the problem differently, because he believed that there is no closed-form solution to the problem. He proposed to find a convergent series as the solution. Weierstrass' idea led to many breakthroughs in the Three-Body problem. In 1892, Heinrich Bruns confirmed Weierstrass' idea. Karl Sundham of Finland obtained the first convergent series for the Three-Body problem in 1913, although the convergence is so slow that it is useless for practical purposes. In 1941, Siegel introduced the so-called *Siegel series* for solutions to a triple collision. In 1991, Qiu Dong Wang also obtained a convergent series for the Three-Body problem. In his solution, he resolved the problem of collisions. When the bodies collide, their trajectories are affected, and this prevents the series from converging. Wang introduced a measure of time that runs faster as two or more bodies approach each other, so that the collision occurs only at infinite time. However, the convergence is again too slow to be useful.

In the 1960s, Stephen Smale of the University of California, Berkeley, proposed a new way to think about the Three-Body problem. He suggested that the dynamical system can be understood in terms of topological transformations.

Chenciner and Montgomery [15] discovered a strange new periodic solution of the Three-Body problem. In their new solution, the three bodies have equal masses and they move along a planar figure-8 curve, which is the same for all three bodies. The three bodies have the same period, but with phase delays of one-third period. The total angular momentum of the system is zero. Except for the well-known case of the triangular circular orbits, the figure-8 orbit is the first periodic solution with this remarkable triple overlap property.

The above is a very brief introduction to the history of the N-Body problem. For further details, we refer to [14] and [55].

1.5 Organization of the thesis

In this thesis, we focus on the CR3BP. AUTO is used to compute periodic solutions of the CR3BP for the cases of the Sun-Earth, the Sun-Jupiter, and the Earth-Moon system. Bifurcation diagrams are constructed from these solutions. To better understand the solutions, PLAUT04 is developed, with which we can view changes in solutions as well as the actual motion of the three bodies in a 3D graphics environment. We hope that this tool will be useful to researchers in Celestial Mechanical and in other fields.

This thesis is organized as follows. The modeling of the Three-Body problem is discussed in Chapter 2. Chapter 3 explains the continuation method used in solving the CR3BP with AUTO, while Chapter 4 presents the results of our computations. A detailed discussion of solution families and bifurcation diagrams of the systems is provided in Chapter 4. The design and development of the new AUTO data visualization package, PLAUT04, is explained in Chapter 5. Appendix A explains how to use AUTO to compute periodic solutions of the CR3BP. Appendix B documents the use of PLAUT04. Appendix C describes some new AUTO commands.

Chapter 2

The Mathematical Description of the CR3BP

2.1 The Two-Body problem

As the legend goes, when the apple dropped on Newton's head, it led to the following insight: any two objects in the universe attract each other with a force that is proportional to the products of their masses and inversely proportional to the square of the distance between the two objects. If we write this expression in mathematical notation, we have

$$F_g = G \frac{m_1 \cdot m_2}{r^2} \quad , \tag{1}$$

where,

F_g is the gravitational force, N,

G is the universal gravitational constant, $G = 6.672 \times 10^{-11} N \cdot m^2/kg^2$,

m_1, m_2 are the masses of the two bodies, kg,

r is the distance between the two bodies, m.

If the direction of the force is taken into account, then the above equation becomes

$$\mathbf{F}_g = G \frac{m_1 \cdot m_2}{||\mathbf{r}||^3} \mathbf{r} \quad , \tag{2}$$

where $\mathbf{r}$ is the vector from m_1 to m_2. So $||\mathbf{r}||$ represents the distance between the two bodies. This is *Newton's universal force law* or *Newton's gravitational law.* In part due to Newton's great discovery, spacecraft can now travel in space, land on the Moon,

and explore Mars. Neil Armstrong's walking on the Moon, the Spirit's successful landing on Mars, and the Cassini-Huygens spacecraft arrival at Saturn on June 30, *etc.*, can be viewed as consequences of Newton's contribution. Newton's universal force law is the mathematical description of the classical Two-Body problem. The Two-Body problem is an elementary one in orbital mechanics, and it is perhaps the first to be described mathematically. It is also the one of the few that can be solved analytically. A detailed description of the solution of the Two-Body problem can be found in [8].

2.2 The Three-Body problem

The Three-Body problem also results from Newton's laws. The equations of the Three-Body problem have many forms. Below we give the most general one.

2.2.1 The general Three-Body problem

In the general Three-Body problem, each of the three bodies has mass m_i, and its position in space is denoted by $\mathbf{r_i}, i = 1, 2, 3$.

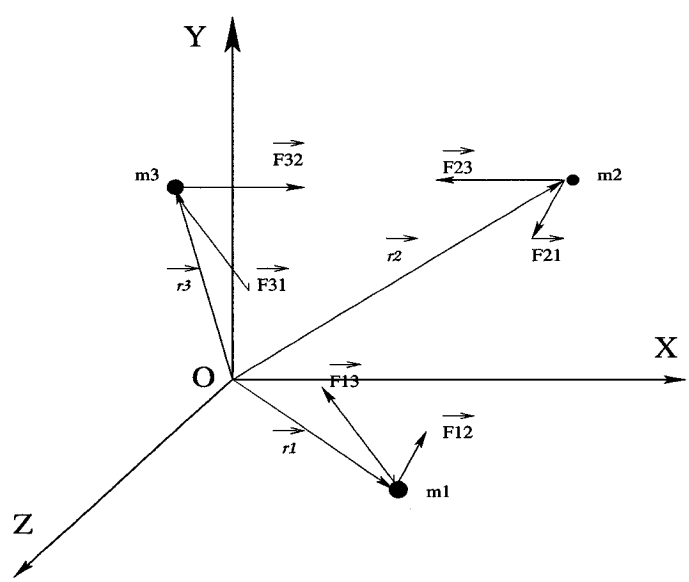


Figure 1: The Three-Body problem

Figure 1 represents the coordinate system of the Three-Body problem. The motion of a body in this system obeys the classical *Newtonian law of inertia*, namely,

acceleration = *force/mass*. Using this law, we obtain,

$$\begin{cases} d^2\mathbf{r}_1/dt^2 &= (\mathbf{F}_{1,2}+\mathbf{F}_{1,3})/m_1 \quad , \\ d^2\mathbf{r}_2/dt^2 &= (\mathbf{F}_{2,1}+\mathbf{F}_{2,3})/m_2 \quad , \\ d^2\mathbf{r}_3/dt^2 &= (\mathbf{F}_{3,1}+\mathbf{F}_{3,2})/m_3 \quad . \end{cases} \tag{3}$$

Here,

m_i is the mass of the object i,

$\mathbf{r}_i$ is the position vector of the object m_i in space,

$\mathbf{F}_{i,j}$ is the gravitational force of attraction of object m_i toward object m_j,

t is the time.

From geometry, we know that, $\mathbf{r}_{ij} = (\mathbf{r}_j - \mathbf{r}_i)$. According to Newton's universal force law, if we don't consider the direction of the forces, we have,

$$\|\mathbf{F}_{ij}\| = G\,\frac{m_i \cdot m_j}{\|\mathbf{r}_{ij}\|^2} \quad . \tag{4}$$

We know that $\mathbf{F}_{ij}$ is in the direction of the unit vector $\mathbf{r}_{ij}/\|\mathbf{r}_{ij}\|$. Hence, the vector form of Equation (4) is

$$\mathbf{F}_{ij} = G\,\frac{m_i \cdot m_j \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^3} \quad . \tag{5}$$

Substituting Equation (5) into Equation (3), we get the classical form of motion of the Three-Body problem, namely,

$$\frac{d^2\mathbf{r}_i}{dt^2} = G\left(\frac{m_j \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^3} + \frac{m_k \cdot \mathbf{r}_{ik}}{\|\mathbf{r}_{ik}\|^3}\right) \; ; \; i,j,k = 1,2,3 \; ; \; i \neq j, \; i \neq k, \; j \neq k \; . \tag{6}$$

There are also other methods to formulate the Three-Body problem. The most familiar ones are Lagrangian Formulation, the Jacobi Formulation, and the Hamilton and Delaunay Formulation. Detailes of these can be found in [10, 14].

2.2.2 The CR3BP

Although the equations of the general Three-Body problem are simple enough in form, they are difficult to solve. In Celestial Mechanics, the equations have been simplified further. For example, suppose we want to study the motion of a satellite around the Earth and the Moon. The Earth and the Moon have their own motions,

which affect each other. Compared to the Earth and the Moon, the mass of the satellite is so small that its attraction to the Moon and the Earth can be ignored. This assumption is reasonable, given that the mass of the Earth is $5.9742 \times 10^{24} kg$, the mass of the Moon is $7.35 \times 10^{22} kg$, whereas a satellite weighs at most several tons. More precisely, in the *restricted Three-Body problem*, the mass of the third body is taken to be small enough so that it does not influence the motion of the other two bodies. The two big bodies, called *primaries*, move under their mutual attraction and have an ordinary Two-Body motion. We want to determine the movement of the third body for given initial conditions, and given the motions of the primaries. The orbits of the two primaries can be circular, elliptic, parabolic, hyperbolic. The movement of the third body can be linear, planar or in three-dimensional space. Based on different assumptions on the orbits of the two primaries and the type of motion of the third body, many possible forms of the restricted Three-Body problems arise [82]. The most commonly considered is the *circular restricted Three-Body problem (CR3BP)*, where one of the two primaries moves in a planar circular orbits around the other. This thesis focuses on this special case only.

Based on the above assumptions, and using Equations (6), the movement of the third body in the CR3BP is determined by the equation

$$\frac{d^2\mathbf{r}_3}{dt^2} = G\left(\frac{m_1 \cdot \mathbf{r}_{31}}{\|\mathbf{r}_{31}\|^3} + \frac{m_2 \cdot \mathbf{r}_{32}}{\|\mathbf{r}_{32}\|^3}\right) . \tag{7}$$

In the CR3BP, the two primaries move in circular planar orbits. Thus it is natural to consider the system in a rotating coordinate frame. In a rotating frame, the two primaries are considered as fixed. The origin of the frame is selected at the barycenter of the two primaries, and the x-axis points from the large primary to the small primary. The z-axis is orthogonal to the orbital plane, and the y-axis completes the right-handed orthogonal coordinate system, as shown in Figure 2. The x, y axes rotate with constant angular velocity, ω. The two primaries are fixed at $(x_1, 0, 0)$ and $(x_2, 0, 0)$, where x_1 is negative, and the third body, called the *infinitesimal*, is located at (x, y, z).

If we use x' to represent the first order derivative of x, and use x'' to represent the

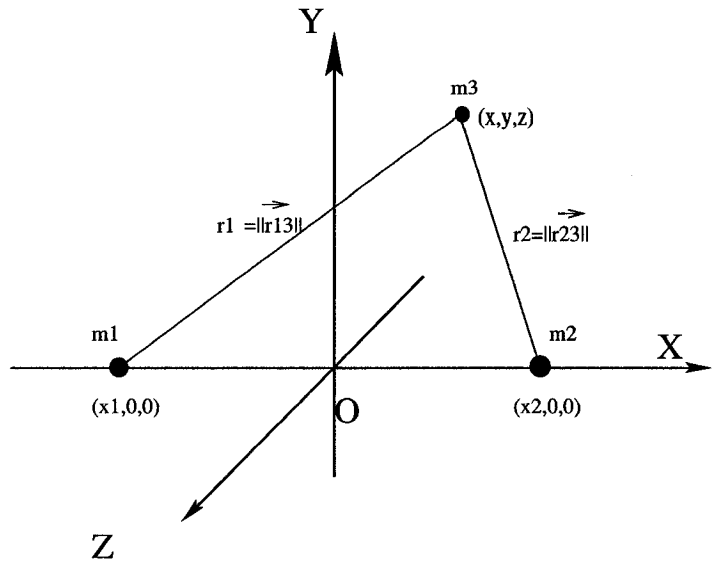


Figure 2: The Three-Body problem in a rotating frame

second order derivative of x, from Equation (7), we get

$$\begin{aligned} x'' &= G\left(\frac{m_2(x_1-x)}{\|\mathbf{r}_{31}\|^3} + \frac{m_1(x_2-x)}{\|\mathbf{r}_{32}\|^3}\right) + 2\omega y' + \omega^2 x \quad , \\ y'' &= -Gy\left(\frac{m_1}{\|\mathbf{r}_{31}\|^3} + \frac{m_2}{\|\mathbf{r}_{32}\|^3}\right) + 2\omega x' + \omega^2 y \quad , \\ z'' &= -Gz\left(\frac{m_1}{\|\mathbf{r}_{31}\|^3} + \frac{m_2}{\|\mathbf{r}_{32}\|^3}\right) \quad . \end{aligned} \tag{8}$$

Here, ω is the rotation speed of the frame, determined by

$$\omega^2\|\mathbf{r}_{12}\|^3 = G\,(m_1+m_2) \quad , \tag{9}$$

and

$$\begin{aligned} \|\mathbf{r}_{12}\| &= x_2 - x_1 \quad , \\ x_1 &= -\frac{m_1\cdot\|\mathbf{r}_{12}\|}{m_1+m_2} \quad , \\ x_2 &= \frac{m_2\cdot\|\mathbf{r}_{12}\|}{m_1+m_2} \quad . \end{aligned} \tag{10}$$

To simplify the system further, a parameter μ is introduced, which represents the ratio of the mass of the small primary to the total mass of the system. For example,

for the Earth-Moon system, $\mu = 0.01215$; for the Sun-Jupiter system, $\mu = 9.53 \times 10^{-4}$; for the Sun-Earth system, $\mu = 3.0 \times 10^{-6}$. The units are chosen so that the distance between the primaries, the sum of the masses of the primaries, and the angular velocity ω of the primaries are all equal to one. The axes rotate with constant angular velocity, ω, and the small primary is located at $(\mu, 0, 0)$ and the large primary at $(1-\mu, 0, 0)$. We can choose dimensions so that the gravitational constant G is equal to one. Then the orbital period is

$$T = 2\pi. \tag{11}$$

Equation (8) can then be rewritten [19, 74] in the following form

$$\begin{aligned} x'' &= 2y' + x - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} \ , \\ y'' &= -2x' + y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \ , \\ z'' &= -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3} \ , \end{aligned} \tag{12}$$

where

$$\begin{aligned} r_1 &= \|\mathbf{r}_{31}\| = \sqrt{(x+\mu)^2 + y^2 + z^2} \ , \\ r_2 &= \|\mathbf{r}_{32}\| = \sqrt{(x-1+\mu)^2 + y^2 + z^2} \ . \end{aligned} \tag{13}$$

The above equations have an integral of motion, which is a function of the coordinates and the velocities, and which is constant along a trajectory in phase space, namely, the energy (or *Jacobi constant*) [19, 74],

$$E = \frac{x'^2 + y'^2 + z'^2}{2} - U(x, y, z) - \frac{\mu(1-\mu)}{2} \ , \tag{14}$$

where

$$U = \frac{1}{2(x^2+y^2)} + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} \ . \tag{15}$$

The Jacobi's integral is the only integral of the restricted circular Three-Body problem [19].

2.2.3 Some characteristics of the CR3BP

Equation (12) has several characteristics. This section gives a summary [14, 19, 30].

- There are five equilibrium points, also called the *libration points* or *Lagrange points*, L1, L2, L3, L4, and L5. All five lie in the same plane as the primaries. L1, L2, and L3 are on the same line as the two primaries, and L4 and L5, each form an equilateral triangle with the primaries.

- When $\mu = 0$, the small primary does not influence the large primary. If $\mu = 0.5$, the two primaries have the same mass. The case $\mu = 0$, is called the Kepler limit. The case $\mu \to 0$ is called the Hill limit. The case $\mu = 0.5$ is called the Copenhagen problem.

- The phase space has an invariant subspace of planar orbits, for which $z = z' = 0$. There are also symmetries: first, if (x, y, z) is a solution, then so is $(x, y, -z)$. Second, if (x, y, z) is a solution then so is $(x, -y, z)$, provided time is reversed also.

- When $\mu = 0$, there is an infinite number of libration points. These are located on a unit circle. In this special situation, the small primary which has zero mass, is also located on the circle.

Many of the great mathematicians, such as Euler, Lagrange, Jacobi, Hill, Poincaré, Levi-Civita, and Birkhoff, have studied the CR3BP. The method of simplifying the equations by introducing a rotating coordinate system, as done above, was first used by Euler. Jacobi discovered the integral of motion, now called the Jacobi integral, in this coordinate system. Poincaré made great contributions to the qualitative theory of Celestial Mechanics. Birkhoff further developed the qualitative methods. Sundman proved the existence of an infinite series solution in the whole plane to the Three-Body problem, using analytical methods of Tullio Levi-Civita and Paul Painlevé. Since such global solutions are available for this problem, the restricted problem of three bodies can be considered to be completely "solved". However, this "solution" does not address issues of stability and allowed regions of motion. Furthermore, to get even a moderately accurate result, a large number of terms is needed, and so this "solution" does not have much practical use [82].

2.3 The general N-Body problem

Figure 3 depicts an N-Body system.

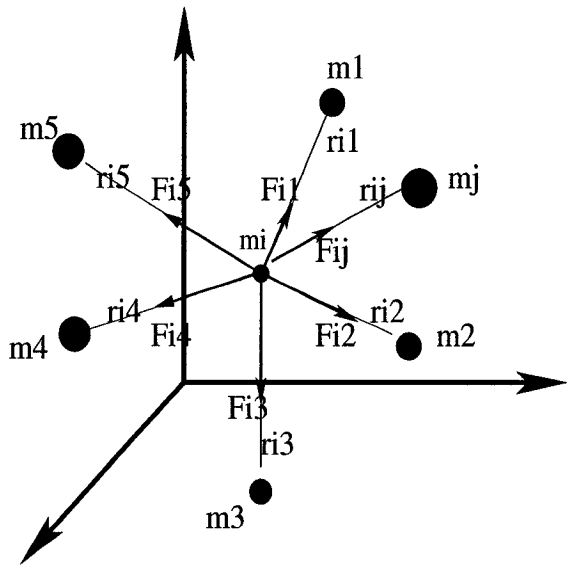


Figure 3: The N-Body problem

The equation of motion for n particles of mass $m_i, i = 1, 2, ..., n$ can be written as follows [65],

$$\mathbf{F}_i = G \sum_{j \neq i,\ j=1}^{n} \frac{m_i \cdot m_j \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^3}, \quad i = 1, \cdots, n \ . \tag{16}$$

This is the mathematical equation for the so-called N-Body problem.

This thesis is focused on the CR3BP, so no further discussion of the general N-Body problem will be given. To learn more about the general N-Body problem, see, for example, [65, 78, 79].

2.4 The surfaces of zero relative velocity of the CR3BP

The velocity, v, of the infinitesimal satisfies $v^2 = x'^2 + y'^2 + z'^2$. By substituting the velocity into Equation (14), we obtain

$$v^2 = x^2 + y^2 + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - C \ , \tag{17}$$

where C is a constant.

Now let $v = 0$. Then Equation (17) becomes

$$0 = x^2 + y^2 + 2\frac{(1-\mu)}{r_1} + \frac{2\mu}{r_2} - C \quad . \tag{18}$$

This defines *Hill's limiting surface*, which represents the surface of zero relative velocity for the CR3BP. It is named after George William Hill, who used these surfaces to prove that the Earth-Moon distance has an upper bound for all time. Although no information about the orbits of the infinitesimal within space can be drawn from it, the equation is useful to discuss general properties of the CR3BP.

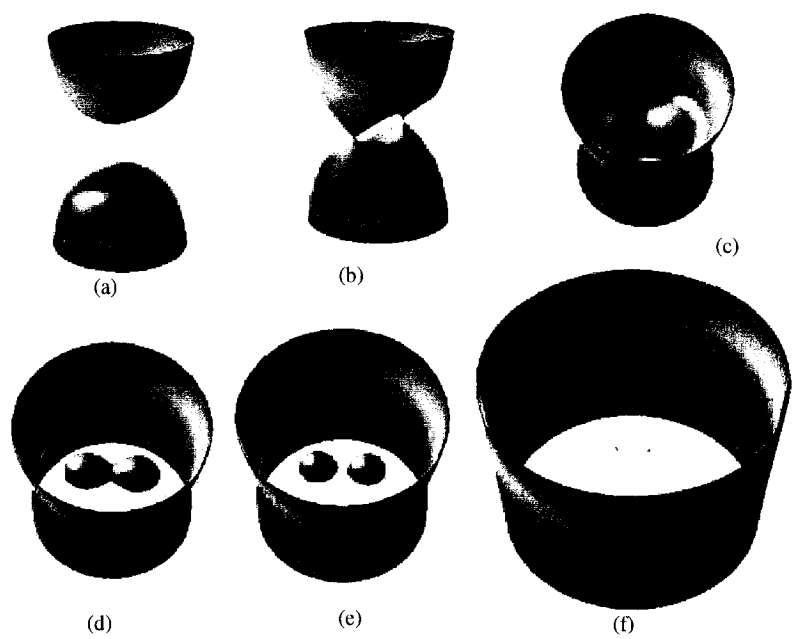


Figure 4: The zero velocity surface

Figure 4 [64] shows diagrams depicting Hill's limiting surface. Hill's limiting surface [63] has the following properties,

1. If both C and $(x^2 + y^2)$ are large, then the equation becomes $x^2 + y^2 \approx C$. It represents a circle in the x-y plane. In the three dimensional x-y-z space, it is a cylinder which is perpendicular to the x-y plane. This cylinder shrinks to form a waist, when z approaches zero. If C is large enough and either r_1, or r_2 is very small, the equation describes two separate ovals. The big oval encloses the large primary and the small oval encloses the small primary (see Figure 4(c)). The

area between the cylinder and ovals has imaginary velocity, and is inaccessible to the infinitesimal. It is called *the forbidden zone.* In the cases of Figure 4(e), because the forbidden zone separates the three regions, if an infinitesimal starts off from somewhere except the forbidden zone, it then remains in that region. Thus the motion of an infinitesimal can only take place in the two ovals or outside the cylinder.

2. When C decreases, the size of the inner ovals increases, while the outer cylinder shrinks. As a result, at some value $C1$, the inner two ovals meet at some critical point, L1, the first libration point (See Figure 4(d)). At this point an infinitesimal still can not reach the other inner oval, when it starts off within one of them. Since the infinitesimal has zero velocity at L1, it cannot pass the critical point into the other oval. However, with further decrease of C, the two inner ovals merge into one surface. In this case, an infinitesimal can pass from one oval into the other.

3. With further decrease of C to a certain value $C2$, the wall of the outer cylinder meets the surface of the inner smaller oval. This is another critical point, beyond which all three regions are finally connected. We know this point as L2, the second libration point in space (See Figure 4(c)). An infinitesimal can reach any place except the forbidden region, when it starts outside the forbidden region.

4. When the value of C decreases further to $C3$, the cylinder wall meets the larger oval's surface at L3, the third libration point (See Figure 4(b)).

5. With further decrease of C, the projection of the forbidden zone shrinks to a tadpole-like shape, and finally to the two libration points L4/L5 (See Figure 4(a)). After this point, the forbidden region is no longer connected. It consists of two parts, an upper half and a lower half.

The surface of zero velocity gives a classification of where an infinitesimal can go and where it cannot go. This is important in spacecraft design and in the exploration of space.

In the above discussion, we referred to critical C-values. By solving Equation (18), we can obtain these values of C exactly. For example, for the Earth-Moon system the C values of the critical points are $13.8191, 13.1284$, and 2.0254. Figure 5 shows

the dependence of C on μ. From the diagram we can see that with increase of μ, the values of $C1$ and $C2$ decrease sharply when μ is small. As for $C3$, it increases monotonically as μ increases.

2.5 The libration points of the CR3BP

The libration points were discovered by French mathematician Louis Lagrange in 1772, in his gravitational studies of the Three-Body problem: "how a third, small body would orbit around two orbiting large ones". Libration points are locations in space where the gravitational forces and the orbital motion of a body balance each other. These points represent points in space where a particle will remain for all time.

L1, L2, L3, L4, and L5 are used to denote the individual libration points. At the libration points, the gravitational forces which the object feels equal the centripetal force needed to rotate with the other two bodies. An object at one of these points will not move in the plane of rotation. The object will share the same orbital period as the other two bodies around the system's center of mass.

Lagrange found that L1,L2 and L3 are collinear with the axis connecting the two massive bodies, with L1 between them and the other two on the outside. We can get their positions from the following equation, obtained by substituting $z = 0$ and $y = 0$ into Equation (18).

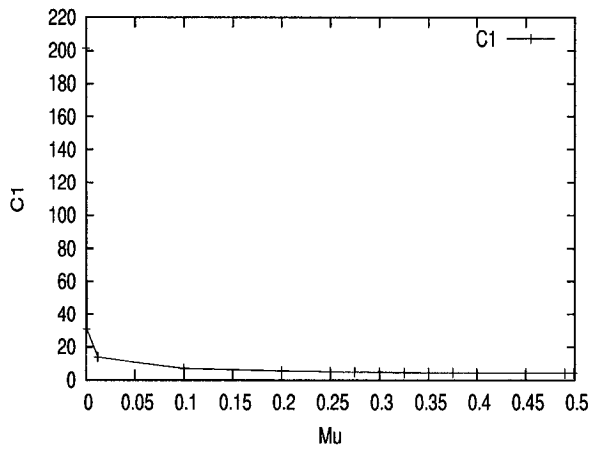
$$0 = x - \frac{(1-\mu)(x-x_1)}{|x-x_1|^3} - \frac{\mu(x-x_2)}{|x-x_2|^3} \quad . \tag{19}$$

For the libration points, L4, L5, the y-coordinate is non-zero. Each of L4 and L5 forms an equilateral triangle with the two primaries. Their positions are

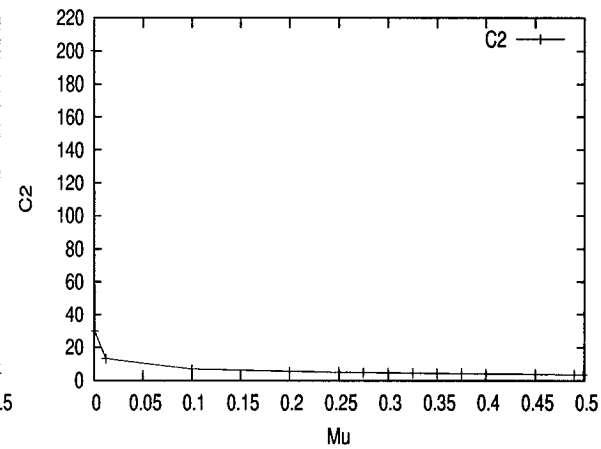
$$(x,\ y,\ z) = (\frac{(1-2\mu)}{2}\ ,\ \pm\frac{\sqrt{3}}{2.0}\ ,\ 0) \quad . \tag{20}$$

Figure 6 [66] shows the position of the libration points for the Sun-Earth system and Figure 7 shows the x and y coordinates of the libration points as a function of μ.

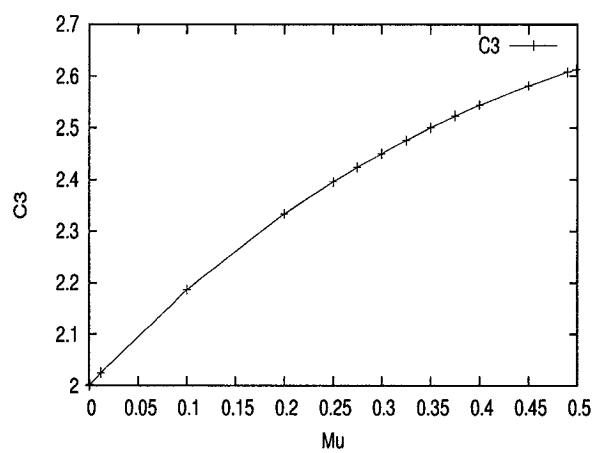
Analytical and computational studies have shown that L1, L2, and L3 are not stable, however L4, and L5 are stable under certain conditions. This result is also observed by scientists. The special properties of the libration points have been used



(a) C_1 versus μ



(b) C_2 versus μ



(c) C_3 versus μ

Figure 5: The dependence of critical Cs on μ

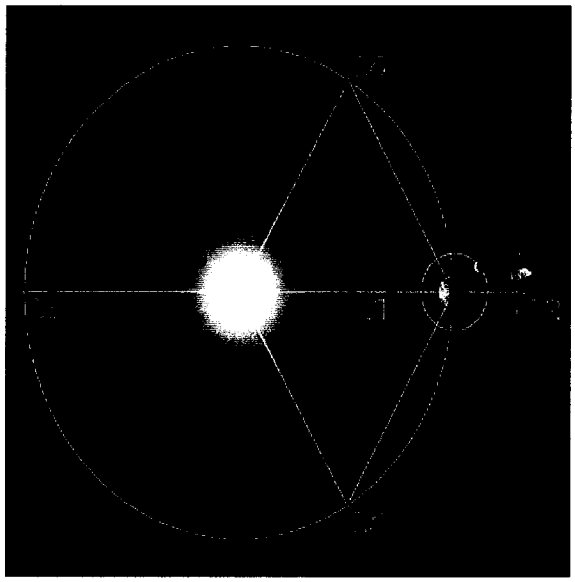


Figure 6: The libration points of the CR3BP

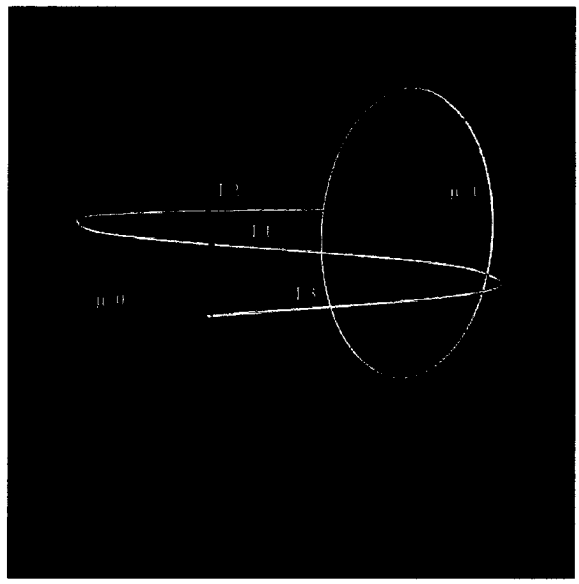


Figure 7: The dependence of the libration points on μ

by some man-made spacecraft. Although Lagrange probably never dreamed of man-made spacecraft being "parked" at the points that bear his name, this is how they are used today. Because of the instability of L1, L2, and L3, a spacecraft at one of these points has to use small rocket firings or other means to stay in the area. An important family of orbits around this area is the "Halo orbits", mentioned in the next paragraphs. We will discuss the Halo orbit families in detail in Chapter 4.

In 1978, the NASA International Sun-Earth Explorer 3 (ISEE-3) became the first spacecraft to orbit the Sun-Earth L1 point, where it traced a Halo orbit [33].

In November 1994, a spacecraft, WIND, was launched toward that position too. It was originally scheduled to be stationed in an orbit about the L1 point by 1996, but later it was sent on an extended mission in a "flower petal" orbit around Earth [70]. Figure 8 [68] shows the WIND petal orbit.

The Solar and Heliospheric Observatory (SOHO), which was launched on December 2, 1995, was also in a Halo orbit around the Sun-Earth L1 position, about a million miles Sun-ward from the Earth [77].

NASA's Genesis Mission, launched on August 8, 2001, was also located at the Sun-Earth system's L1, and it will return to the Earth on September 8, 2004 (see Figure 9 [69]).

In 2001, the Microwave Anisotropy Probe (MAP), was in a quasi-periodic Lissajous orbit around the Sun-Earth L2 position, about a million miles in the opposite direction [67]. Lissajous orbits are the natural motion of a satellite around a collinear libration point in a Two-Body system and require less momentum change to be expended for station keeping than Halo orbits, where the satellite follows a simple circular or elliptical path about the libration point [58].

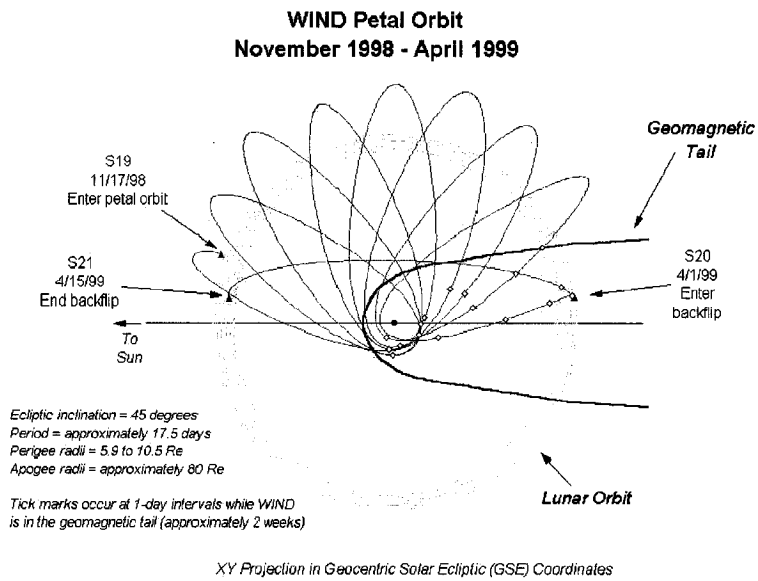


Figure 8: The WIND petal orbit

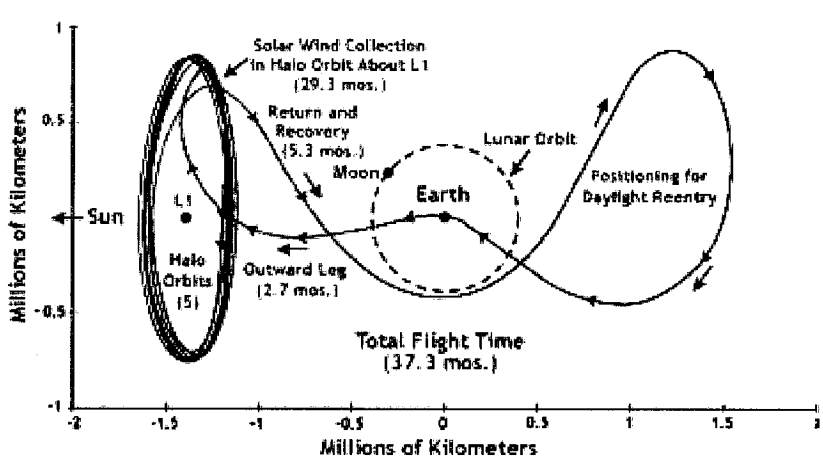


Figure 9: The Genesis Mission

Some other scientific missions that are planned to orbit the Sun-Earth L2 point are NASA's Next Generation Space Telescope (NGST) [36] and ESA's FIRST and

Planck missions [35]. In a more elaborate design, the Terrestrial Planet Finder (TPF) mission will exploit the center manifold of a Halo orbit to fly two spacecrafts to gather information along Lissajous orbits about the Sun-Earth L2 point [6, 37].

The L4 and L5 points are home to stable orbits as long as the mass ratio between the two large masses exceeds 24.96, or equivalently, $\mu < 0.03852$ [17]. The Earth-Sun and the Earth-Moon systems, and many other pairs of bodies in the solar system, satisfy this condition. In the Sun-Earth-Moon system, any object in the Earth-Moon L4 and L5 locations "orbit" the libration point in an 89-day cycle. Objects found orbiting the L4 and L5 points are often called Trojans. There are hundreds of Trojan asteroids in the solar system. Most orbit with Jupiter, but others orbit with Mars. The three large ones that orbit in the L4 and L5 points of the Jupiter-Sun system are called Agamemnon, Achilles and Hector. In 1956 the Polish astronomer Kordylewski discovered large concentrations of dust at the Trojan points of the Earth-Moon system. Recently, the COBE satellite also confirmed the existence of a dust ring following the Earth's orbit around the Sun [18]. One can find a more detailed discussion of the libration points in [19, 74].

2.6 Periodic orbit families near the libration points

The CR3BP has been studied extensively by many scientists for various values of μ. Many details of the solution structure of CR3BP have been computed before. An extensive, yet incomplete list of references appears in [29], where the citations range from Poincaré's *Les Méthodes Nouvelles de la Mécanique Céleste* [72], through the work of Arthur Clarke [16], Farquhar [32], Deprit and Henrard [21], Howell [47], Gómez *et al.* [38, 39, 40], and many others, to the recent work of Martin Lo *et al.* on the Genesis mission [59].

Figure 10 shows a *bifurcation diagram*, which represents schematically various families of periodic orbits for the Earth-Moon system, for which $\mu = 0.01215$.

The Jacobian of Equation (12), evaluated at the libration points L1, L2, and L3, has two pairs of purely imaginary eigenvalues, which give rise to two well-known families of periodic orbits, namely, the planar *Lyapunov orbits*, **L**1, **L**2, and **L**3, and the *Vertical orbits*, **V**1, **V**2, and **V**3, respectively. The Jacobian at the libration points **L**4 and **L**5 has at least one pair of purely imaginary eigenvalues, for all μ, which

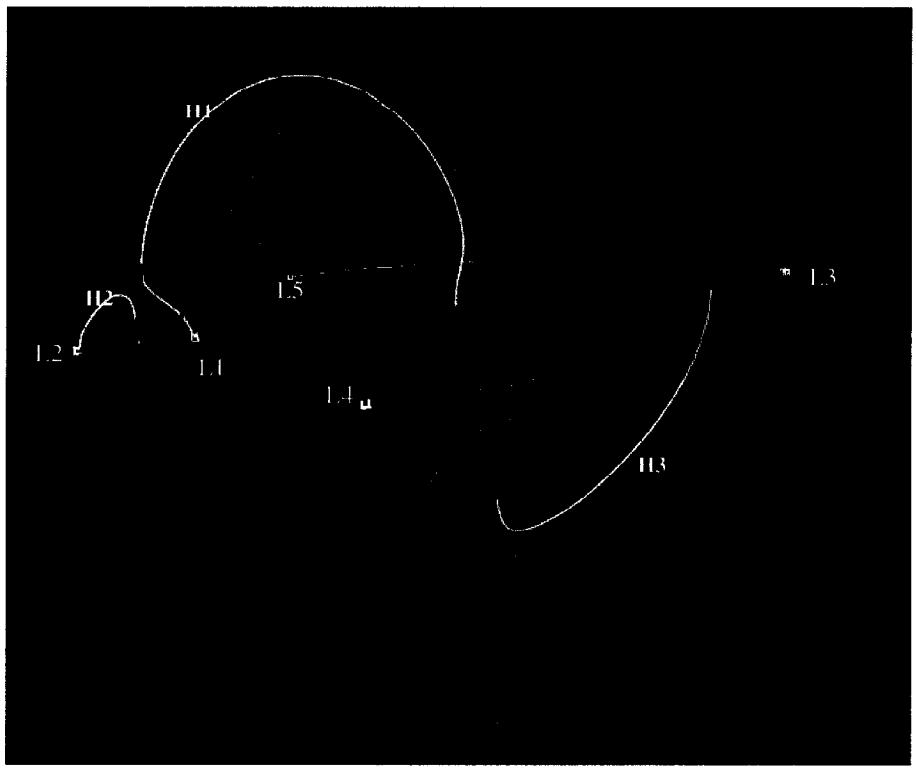


Figure 10: A bifurcation diagram

gives rise to the families **V**4 and **V**5 of Vertical orbits from L4 and L5, respectively. For μ less than the critical value $\mu_1 \equiv 1/2 - \sqrt{23/108} \approx 0.03852$, the Jacobian at the libration point L4 (and L5) has an additional two pairs of purely imaginary eigenvalues, that give rise to two families of planar orbits, that emanate from L4 (and L5) [49].

The bifurcation diagram and its solution families, will be discussed in detail in Chapter 4. In next chapter, we discuss the computation of periodic solutions of the CR3BP using AUTO.

Chapter 3

Numerical Methods

3.1 Continuation

Analytical methods based on perturbation theory, and numerical methods are the two main tools for quantitative analysis. In this thesis, we focus on numerical methods. More information on analytical methods can be found in many books, especially those of Hagihara [43], V. Szebehely [82], and F. Tisserand [84].

Numerical continuation enables the computation of solution manifolds (solution families). There are several software packages for the numerical analysis of bifurcations in dynamical systems. Most are for the computation of one-dimensional solution manifolds. AUTO is one of the earliest packages and perhaps the most widely used. Other software is described in [1, 7, 26, 27, 75, 80], and [56]. Henderson has developed numerical continuation algorithms for higher-dimensional manifold [44]. One can also find more information in [9].

AUTO is used for all calculations in this thesis. In this chapter, we show how AUTO is used to compute periodic solutions of the CR3BP.

3.1.1 The Implicit Function Theorem

The *Implicit Function Theorem ("IFT")* is an indispensable tool in bifurcation analysis. It is also the foundation of numerical continuation [42, 61].

Let $\mathbf{F}\colon \mathbb{R}^{n+1} \to \mathbb{R}^n$ be a smooth function. Let $\mathbf{F}_X(X_0)$ denote the Jacobian matrix of $\mathbf{F}(X)$ evaluated at X_0. Recall that the elements of $\mathbf{F}_X(X_0)$ are the derivatives of the n component functions of $\mathbf{F}$ with respect to the $n+1$ variables X. Note that the

matrix $\mathbf{F}_X(X_0)$ has n rows and $n+1$ columns.

The IFT states that if

$$Rank(F_X(X_0)) \;=\; n \;, \tag{21}$$

then there exists a unique family $X(s)$, and $\delta > 0$, such that

$$\begin{aligned} X(0) &= X_0 \;, \\ F(X(s)) &= 0 \quad \text{for} \quad |s| \leq \delta \;. \end{aligned} \tag{22}$$

3.1.2 Numerical continuation

.

Given the finite-dimensional equation

$$\mathbf{F}(X) \;=\; \mathbf{0} \;, \qquad \mathbf{F} \;:\; \mathbb{R}^{n+1} \;\rightarrow\; \mathbb{R}^n \;, \tag{23}$$

where $\mathbf{F}$ is assumed to be sufficiently smooth. This equation has one more variable than it has equations. Given a solution X_0, if the assumptions of the IFT are satisfied then there is a locally unique *branch* of solutions that passes through X_0 for this system. To compute a next point, say, X_1, on this branch, Newton's method can be used to solve the extended system

$$\begin{aligned} &\text{a)}\;\; \mathbf{F}(X_1) &&= \;\; 0 \;, \\ &\text{b)}\;\; (X_1 - X_0)^* \; \dot{X}_0 &&= \;\; \Delta s \;. \end{aligned} \tag{24}$$

Here $\dot{X}_0$ is the unit tangent to the path of solutions at X_0, the superscript $*$ denotes transpose, and Δs is a scalar, which is the step size in the continuation procedure. The vector $\dot{X}_0$ is also a null vector of the Jacobian matrix $F_X(X_0)$, and can be computed at little cost [27]. Figure 11 shows a geometrical interpretation of this well-known method, known as Keller's *pseudo-arclength method* [53]. The size of the pseudo-arclength step size Δs is normally adapted along the branch depending on the convergence history of Newton's method.

It can be shown that Keller's method works near a *regular solution* X_0, *i.e.*, if the null space of $F_X(X_0)$ is one-dimensional, as in the IFT. In fact, in this case the

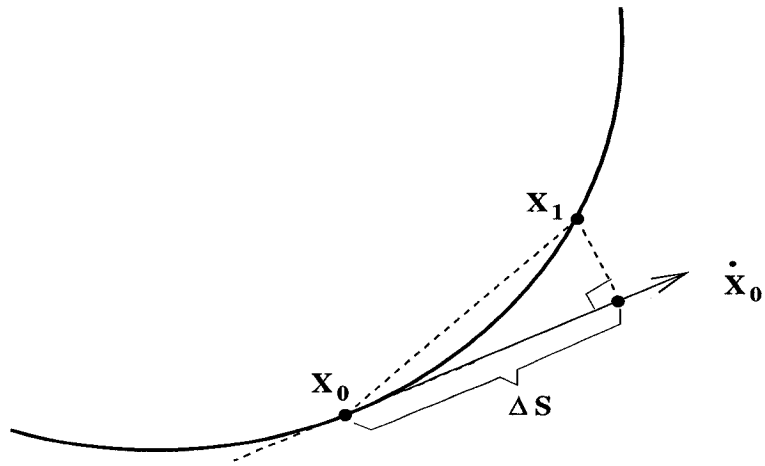


Figure 11: Graphical interpretation of the pseudo-arclength method

Jacobian of Equation (24) evaluated at X_0, *i.e.*, the n+1 by n+1 matrix

$$\begin{pmatrix} \mathbf{F}_X(X_0) \\ \dot{X}_0^{\,*} \end{pmatrix}, \tag{25}$$

is easily seen to be nonsingular. By the Implicit Function Theorem, this guarantees the existence of a locally unique solution branch through X_0. This branch can be parametrized locally by Δs. Moreover, for Δs sufficiently small, and for sufficiently accurate initial approximation (*e.g.*, $X_1^{(0)} = X_0 + \Delta s \dot{X}_0$), Newton's method for solving Equation (24) can be shown to converge. Bifurcation points along the solution branch correspond to singularities of the Jacobian matrix of Equation (25); such points can be located accurately, and there are standard algorithms for switching branches there. These algorithms are implemented in AUTO [26].

3.1.3 Periodic solutions

Pseudo-arclength continuation can also be used to compute a family of periodic solutions of a dynamical system. Given the system

$$\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), \lambda) \ , \qquad \mathbf{f} \ : \ \mathbb{R}^n \times \mathbb{R} \ \rightarrow \ \mathbb{R}^n \ , \tag{26}$$

where the nonlinear function $\mathbf{f}$ consists of n components functions, $f_1(\mathbf{x}, \lambda), \ldots, f_n(\mathbf{x}, \lambda)$; $\mathbf{x} \in \mathbb{R}^n$ is a vector with n components, $x_1, \ldots, x_n$; $\lambda \in \mathbb{R}$ is a physical parameter.

In this case the continuation step corresponding to Equation (24) takes the form of the following constrained periodic boundary value problem [29]:

$$
\begin{aligned}
&\mathrm{a}_1) \quad \mathbf{x}_1'(\mathrm{t}) = \mathrm{T}_1 \mathbf{f}(\mathbf{x}_1(\mathrm{t}), \lambda_1), \\
&\mathrm{a}_2) \quad \mathbf{x}_1(0) = \mathbf{x}_1(1), \\
&\mathrm{a}_3) \quad \textstyle\int_0^1 \mathbf{x}_1(\tau)^* \, \mathbf{x}_0'(\tau) \, \mathrm{d}\tau = 0, \\
\mathrm{b}) \quad &\textstyle\int_0^1 (\mathbf{x}_1(\tau) - \mathbf{x}_0(\tau))^* \dot{\mathbf{x}}_0(\tau) \, d\tau + (T_1 - T_0)\dot{T}_0 + (\lambda_1 - \lambda_0)\dot{\lambda}_0 = \Delta s.
\end{aligned}
\tag{27}
$$

This equation must be solved for $\mathbf{X}_1 = (\mathbf{x}_1(\cdot), T_1, \lambda_1)$, given a solution $\mathbf{X}_0 = (\mathbf{x}_0(\cdot), T_0, \lambda_0)$ and the path tangent $\dot{\mathbf{X}}_0 = (\dot{\mathbf{x}}_0(\cdot), \dot{T}_0, \dot{\lambda}_0)$. Here $T_1 \in \mathbb{R}$ is the unknown period. Equation ($27\mathrm{a}_2$) imposes unit periodicity, after rescaling of the independent variable t. Equation ($27\mathrm{a}_3$) is a phase condition, which fixes the phase of the new orbit $\mathbf{x}_1(\cdot)$ relative to the given orbit $\mathbf{x}_0(\cdot)$. It may be replaced by the classical Poincaré phase condition

$$\mathrm{a}_3') \quad (\mathbf{x}_1(0) - \mathbf{x}_0(0))^* \, \mathbf{x}_0'(0) = 0.$$

However, the integral phase condition ($27\mathrm{a}_3$) has the desirable property of minimizing phase drift relative to $\mathbf{x}_0(\cdot)$, which often allows much bigger continuation steps to be taken [27]. Equation (27b) is the functional form of the pseudo-arclength constraint Equation (24). More details on this boundary value approach for computing periodic solutions can be found in [27]; further references include [25, 52].

In each continuation step, Equation (27) is solved by numerical boundary value algorithms. In particular, AUTO uses piecewise polynomial collocation with Gauss-Legendre collocation points ("*orthogonal collocation*") [5, 20], similar to COLSYS [3], and COLDAE [4], with adaptive mesh selection [76]. Combined with continuation, this allows the numerical solution of "difficult" problems. AUTO determines the *characteristic multipliers* (or *Floquet Multipliers*), that determine asymptotic stability and bifurcation properties, as a by-product of the decomposition of the Jacobian of the boundary value collocation system [27, 34, 57].

3.1.4 Periodic solutions of the CR3BP

In the CR3BP system, there is one conserved quantity, namely, the Jacobi constant E, which is defined in Equation (14).

In order to use AUTO to continue periodic solutions of the CR3BP, we need to modify the original form of the equations. We first rewrite the three-dimensional second-order equations to a standard six-dimensional first-order system, and scale time, so that the period T appears explicitly in the equations. To decrease the order of the system, the velocities v_x, v_y, v_z in directions x, y, z are the appropriate choice to use. We also add periodic boundary conditions and the integral phase constraint, as in Equation (27).

The second change we made is that, in order to use boundary value algorithms, we introduce an *unfolding term* with corresponding *unfolding parameter*, as discussed in detail in [60] and [29]. A suitable choice for the unfolding term is $\lambda \nabla E$, where λ is the unfolding parameter. However, this choice is not unique, and for the CR3BP it is more convenient to introduce a simpler unfolding term, namely, one that corresponds to "damping".

After we introduce the unfolding parameter, the first order system then becomes:

$$\begin{aligned}
x' &= Tv_x \quad , \\
y' &= Tv_y \quad , \\
z' &= Tv_z \quad , \\
v_x' &= T[2v_y + x - \tfrac{(1-\mu)(x+\mu)}{r_1^3} - \tfrac{\mu(x-1+\mu)}{r_2^3}] + \lambda v_x \quad , \\
v_y' &= T[-2v_x + y - \tfrac{(1-\mu)y}{r_1^3} - \tfrac{\mu y}{r_2^3}] + \lambda v_y \quad , \\
v_z' &= T[\tfrac{-(1-\mu)z}{r_1^3} - \tfrac{\mu z}{r_2^3}] + \lambda v_z \quad .
\end{aligned} \tag{28}$$

It is important to stress that, while λ is a scalar unknown that is solved for at each continuation step, its value will be zero (up to numerical precision) upon convergence of Newton's method. This simple, yet crucial observation is discussed in detail in [29], and theoretically justified in [60].

3.1.5 Locating the libration points

The CR3BP has five libration points, for each specific value of μ with $0 < \mu < 1.0$. In Chapter 2, an analytical solution of the libration points has been given for L4 and

L5, while for L1, L2, and L3, a simplified equation is given, which can be solved easily using root finders. However, AUTO can use continuation to solve the entire curve of the libration points as a function of μ, as shown in Figure 7.

3.2 Converting data from the rotating frame to the inertial frame

In Chapter 2, we used a rotating coordinate system to simplify the equations of the CR3BP. This formulation decreases the computational complexity. It is a good formulation for quantitative analysis. Nevertheless, to gain an intuitive understanding of the motions, it is helpful to represent the motion of the system also in the inertial frame. In this section, we will discuss how to transform from a rotating frame to an inertial frame [23].

The Sun-Earth system is taken as an example for the discussion. For other systems, the same method can be used.

In Chapter 2, the primary bodies (the Sun and the Earth) are assumed moving in circular orbits around the system's barycenter (center of mass). A rotating frame, centered at the barycenter, is used. The x-axis lies along the line from the large primary (the Sun) to the small primary (the Earth). The z-axis points along the direction normal to the orbital plane. The y-axis completes the coordinate system. The position of a particle in the system, relative to the above coordinate system, is given by (x, y, z) . The velocity is given by (x', y', z') , where the prime denotes a derivative with respect to t, the dimensionless time in the CR3BP coordinates.

The data conversion is for animation purpose only, therefore, some conversions, such as the conversion of mass, *etc.*, are not considered.

3.2.1 Time conversion

The following equation is used to convert the dimensionless time t in Equation (12) to time τ in physical units.

$$t = \frac{2\pi}{T}\tau . \tag{29}$$

However, for CR3BP computations using AUTO, we scale the period to the closed

interval $[0,1]$, so, as in AUTO data, we use

$$t = \frac{1}{T}\tau , \tag{30}$$

where T is the orbit period of the Earth in the physical system. In the general case T represents the period of the small primary rotating around the large primary.

It follows that

$$\begin{pmatrix} dx/d\tau \\ dy/d\tau \\ dz/d\tau \end{pmatrix} = \begin{pmatrix} dt/d\tau \cdot dx/dt \\ dt/d\tau \cdot dy/dt \\ dt/d\tau \cdot dz/dt \end{pmatrix} = \frac{1}{T}\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} . \tag{31}$$

3.2.2 Position conversion

Let $\mathbf{r}_{SE}$ be the position vector of the Earth with respect to the Sun in the inertial frame at a given time. The distance between the primaries is $R_{SE} = \|\mathbf{r}_{SE}\|$. Furthermore, the velocity vector of the Earth with respect to the Sun in the inertial frame can be written as $\mathbf{v}_{SE} = d\mathbf{r}_{SE}/d\tau$. Thus the orbital angular velocity vector is $\mathbf{h} = \mathbf{r}_{SE} \times \mathbf{v}_{SE}$. We also define $H = \|\mathbf{h}\|$.

If $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the unit vectors along the x, y and z axes, respectively, in the rotating frame, then we have

$$\begin{aligned} \mathbf{e}_x &= \mathbf{r}_{SE}/R_{SE} , \\ \mathbf{e}_z &= \mathbf{h}/H , \\ \mathbf{e}_y &= \mathbf{e}_z \times \mathbf{e}_x . \end{aligned} \tag{32}$$

A particle's position in the inertial frame, with coordinates (x, y, z) in the rotating frame, is then determined by

$$\begin{aligned} \mathbf{r} &= \mathbf{r}_{SE} \times \mathbf{P}, \\ \mathbf{P} &= (x-c)\mathbf{e}_x + y\mathbf{e}_y + z\mathbf{e}_z, \end{aligned} \tag{33}$$

where the offset c is used as a parameter for controlling where the coordinate origin

is put, and

$$c = \begin{cases} 1-\mu, & \text{if the center of the inertial system is the Earth ;} \\ -\mu, & \text{if the center is the Sun ;} \\ 0, & \text{if the center is the barycenter.} \end{cases}$$

3.2.3 Velocity conversion

To transform the velocity from the rotating frame to the inertial frame, we must consider not only the change of the coordinates (x, y, z), but also the motion of the axes $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$, and the change of the distance R_{SE} between the primaries.

Because the coordinate axes are orthonormal, if we define the angular velocity vector ω in the rotating frame as

$$\omega = \omega_x \cdot \mathbf{e}_x + \omega_y \cdot \mathbf{e}_y + \omega_z \cdot \mathbf{e}_z, \tag{34}$$

then we have

$$\begin{aligned} d\mathbf{e}_x/d\tau &= \omega \times \mathbf{e}_x \ , \\ d\mathbf{e}_y/d\tau &= \omega \times \mathbf{e}_y \ , \\ d\mathbf{e}_z/d\tau &= \omega \times \mathbf{e}_z \ . \end{aligned} \tag{35}$$

From Equations (29), $\cdots$, (35), we can show that in general

$$\begin{aligned} \omega_x &= R_{SE}(\mathbf{a}_p \cdot \mathbf{e}_z)/H, \\ \omega_y &= 0, \\ \omega_z &= H/R_{SE}^2. \end{aligned} \tag{36}$$

Here, $\mathbf{a}_p$ represents the perturbing acceleration experienced by the Earth, apart from the gravitational acceleration of the Sun. If the orbits of the primaries always remain in a fixed plane, as in the Two-Body problem, then the orbital angular velocity and frame angular velocity are parallel, $\omega = \mathbf{h}/R_{SE}^2 = \omega_z \times \mathbf{e}_z$, where $\omega_z = H/R_{SE}^2$. The average value of ω_z is $2\pi/T$, where T is again the orbital period of the Earth. In fact, in the CR3BP, the Earth's orbit is assumed to be circular, so $\omega_z \equiv 2\pi/T$. The term ω_x can be neglected.

Now the inertial velocity of the infinitesimal can be computed by

$$\mathbf{v} = R_{SE}\left\{\frac{dx}{d\tau}\mathbf{e}_x + \frac{dx}{d\tau}\mathbf{e}_y + \frac{dx}{d\tau}\mathbf{e}_z\right\} + \omega \times \mathbf{r} + \frac{dR_{SE}}{d\tau}\mathbf{P} \ , \tag{37}$$

where

$$\omega \times \mathbf{r} = \omega_z\left\{-y\mathbf{e}_x + (x-c)\mathbf{e}_y\right\} + \omega_x(-z\mathbf{e}_x + y\mathbf{e}_z) \ , \tag{38}$$

and

$$\frac{dR_{SE}}{dt} = \frac{\mathbf{r}_{SE}\cdot\mathbf{v}_{SE}}{R_{SE}} \ . \tag{39}$$

3.2.4 Transforming from the rotating frame to the inertial frame

The algorithm based on the above discussion is as follows.

1. Compute the position and velocity of the Earth with respect to the Sun.

2. Compute the angular velocity vector $\mathbf{h} = \mathbf{r}_{SE} \times \mathbf{v}_{SE}$ and its magnitude $H = |\mathbf{h}|$. If we use the analytical approximation above, then

$$\begin{aligned} \mathbf{h} &= \omega_z R_{SE}^2(0,\ 0,\ 1) \ , \\ H &= \omega_z R_{SE}^2 \ . \end{aligned} \tag{40}$$

3. Compute the unit vectors $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$. In the analytical approximation

$$\begin{aligned} \mathbf{e}_x &= (\cos(\omega_z t),\ \sin(\omega_z t),\ 0) \ , \\ \mathbf{e}_y &= (-\sin(\omega_z t),\ \cos(\omega_z t),\ 0) \ , \\ \mathbf{e}_z &= (0,\ 0,\ 1) \ . \end{aligned} \tag{41}$$

4. Compute the position vector $\mathbf{r}$ using Equation (33).

5. Convert the velocities (x, y, z) to derivatives with respect to t using Equation (31).

6. Compute the velocity vector using Equations (37), (38), and (39).

In analytical approximation, we assume R_{SE} is constant, so $dR_{SE}/d\tau = 0$. The perturbing effects of other bodies is also ignored, so $\omega_x = 0$ in Equation (38).

3.2.5 Examples of orbits in the inertial frame

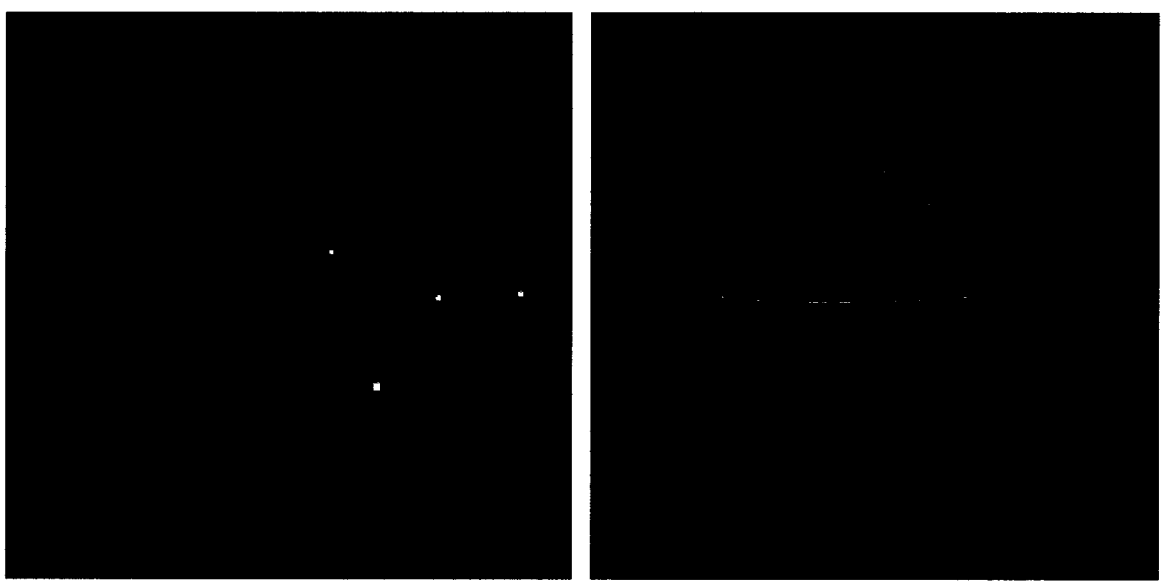
This section gives some examples to show what solutions look like in an inertial frame and in a rotating frame. A more detailed description of solutions will be given in the next chapter.

Figure 12 shows a **V**4 orbit of the Earth-Moon system. Figure 12(a) is the orbit in the rotating frame, and Figure 12(b) shows the orbit in the inertial frame centered at the Moon.

Figure 13 is a **V**2 orbit. Figure 13(a) shows the orbit in the rotating frame. Both the primaries and the libration points are shown in this diagram. Figure 13(b) depicts its projection onto the x-y plane in the inertial frame, and a three dimensional picture is shown in Figure 13(c).

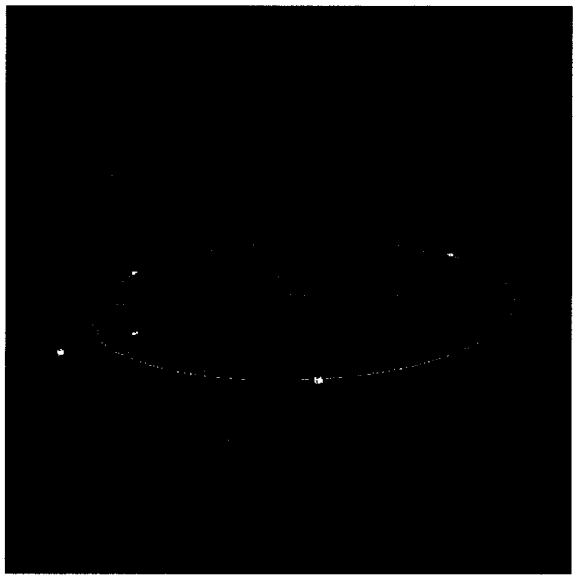
Figure 14 and Figure 16 show two orbits from the **C**2 family of the Earth-Moon system. Note that this is a planar orbit. Figure 15 shows an **A**1 orbit of the Earth-Moon system.

In this chapter, continuation and algorithm for converting data from a rotating frame to an inertial frame were discussed. Their applications to the CR3BP will be discussed in the next chapter.

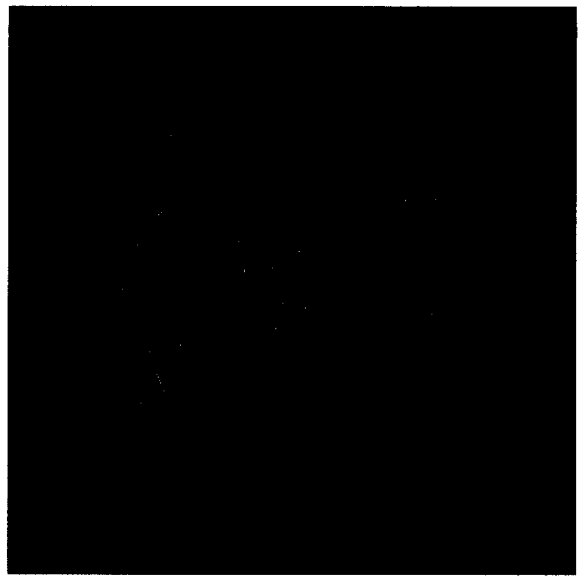


(a) Rotating frame (b) Inertial frame

Figure 12: An Earth-Moon **V**4 orbit

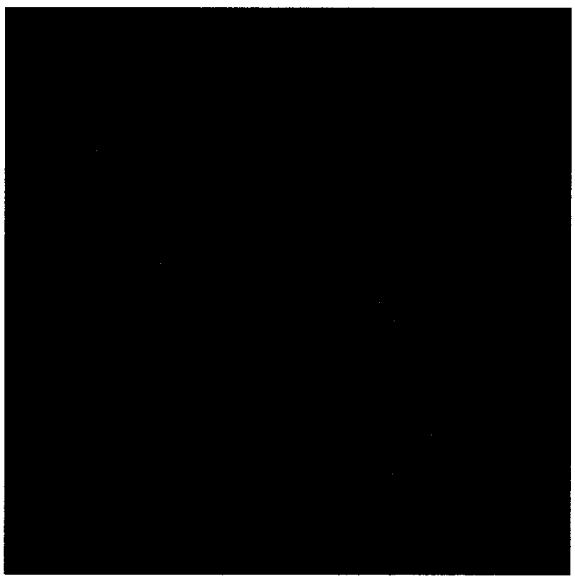


(a) Rotating frame



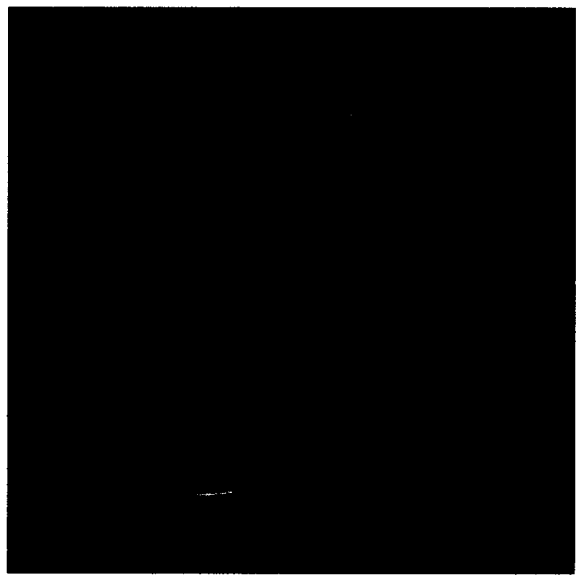
(b) Inertial frame - 2D view

Figure 13: An Earth-Moon **V**2 orbit



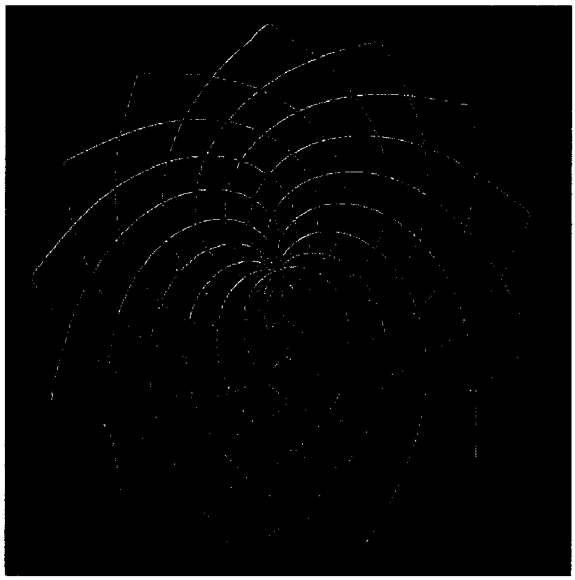
(c) Inertial frame - 3D view

Figure 13: An Earth-Moon **V**2 orbit (cont'd)

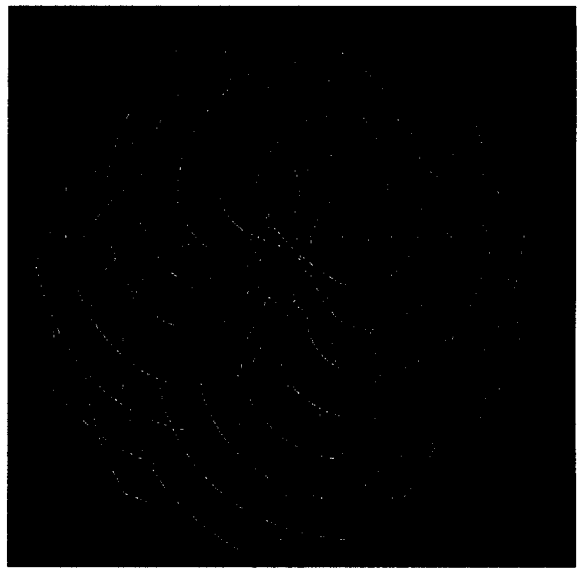


(a) Rotating frame

Figure 14: An Earth-Moon **C**2 orbit

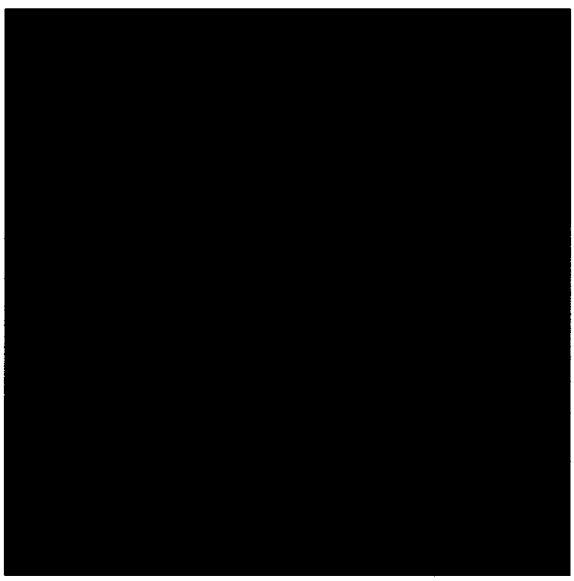


(b) Inertial frame (Earth-centered)

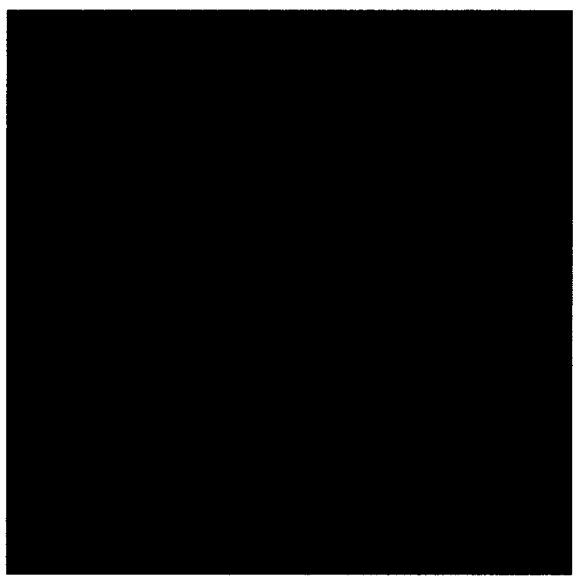


(c) Inertial frame (Sun-centered)

Figure 14: An Earth-Moon **C**2 orbit (cont'd)

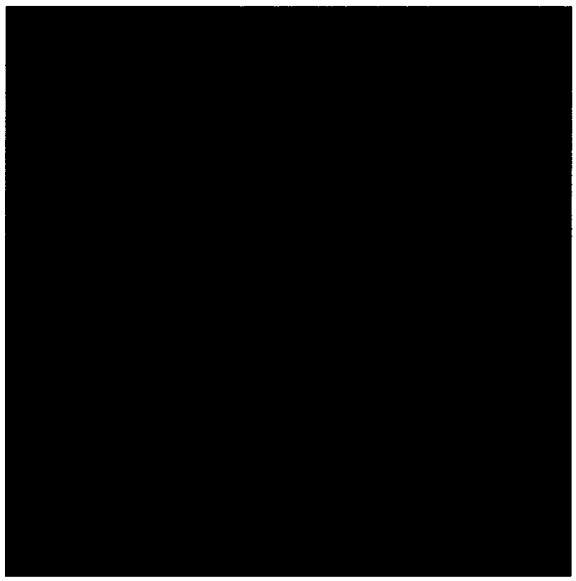


(a) Rotating frame



(b) Inertial frame

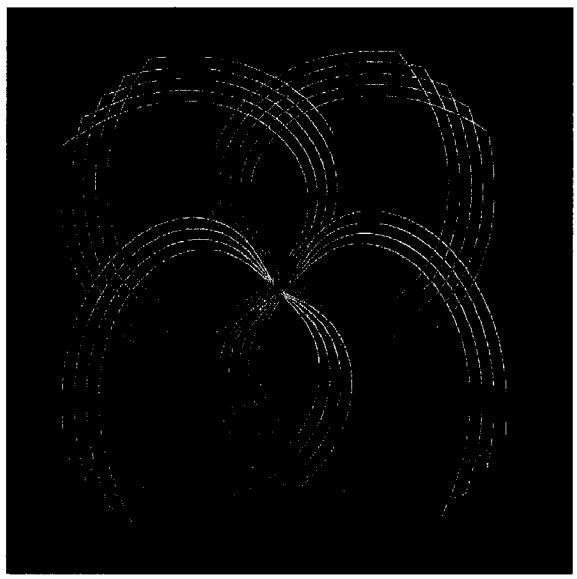
Figure 15: An **A**1 orbit for the Earth-Moon system (Earth-centered)



(a) Rotating frame

(b) Inertial frame (Earth-centered)

Figure 16: Another **C**2 orbit for the Earth-Moon system



(c) Inertial frame (Sun-centered)

Figure 16: Another **C**2 orbit for the Earth-Moon system (cont'd)

Chapter 4

Application of AUTO to the CR3BP

4.1 Introduction

In this chapter, AUTO is used to compute families of periodic solutions for the Earth-Moon, the Sun-Earth, and the Sun-Jupiter system. Bifurcation diagrams for each system are constructed from the results.

In the computation, the mass ratio of 0.01215 is used for the Earth-Moon system, 3.0×10^{-6} is used for the Sun-Earth system, and 9.53×10^{-4} is used for the Sun-Jupiter system.

Gómez and Mondelo [38] computed families of orbits arising from L1, L2, and L3, as well as their bifurcating branches. Ichtiaroglou and Michalodimitrakis [50], Hénon [46], Howell and Campbell [48] also provided useful information on bifurcations of these systems.

Because of the similarity of the three systems, we focus on the Sun-Earth system in our discussion. Differences among the Earth-Moon, the Sun-Earth, and the Sun-Jupiter systems are indicated as needed.

4.2 Objects and annotations

4.2.1 Object representation

The diagrams include several geometric objects, namely, primaries, libration points, branch points of different types, orbits, and a reference plane, which are represented in different ways.

Primaries

Textured spheres are used to represent the primaries. Their sizes are approximately proportional to their "weights", though they are not proportional to the distance between the primaries. In the Earth-Moon system, the large primary is the Earth and the small primary is the Moon. In the other two systems, the large primary is the Sun and the small primary is the Earth, or Jupiter, respectively.

Libration points

White cubes are used to represent the libration points. From the diagram, we can see that three of them, namely, L1, L2, and L3, are collinear with the two primaries. The other two, namely, L4 and L5, form two equilateral triangles with the two primaries, as discussed in previous chapters.

Solutions/orbits

Tubes or curves are used to represent the periodic solutions. Each curve represents a family of periodic solutions in the bifurcation diagram. Surfaces can also be used to represent the manifolds formed by the orbits. Different coloring schemes can be applied to solutions, according to their family, type, speed, energy, time, point number, *etc.* We can also animate solutions, and the motion of the infinitesimal, with PLAUT04.

Bifurcation Points

Bifurcation points are represented as spheres in the bifurcation diagram. In the solution diagram, the corresponding orbits are represented in a distinct color to distinguish them from the other orbits.

Reference Plane

A planar circular disk is used to represent the orbit of the smaller primary around the large primary in the rotating frame. This disk lies in the x-y plane. Any solution family in the bifurcation diagram touching this plane has a planar solution at that point. For example, the curve that represents **L1** lies entirely on the disk, which means that all **L1** orbits are planar.

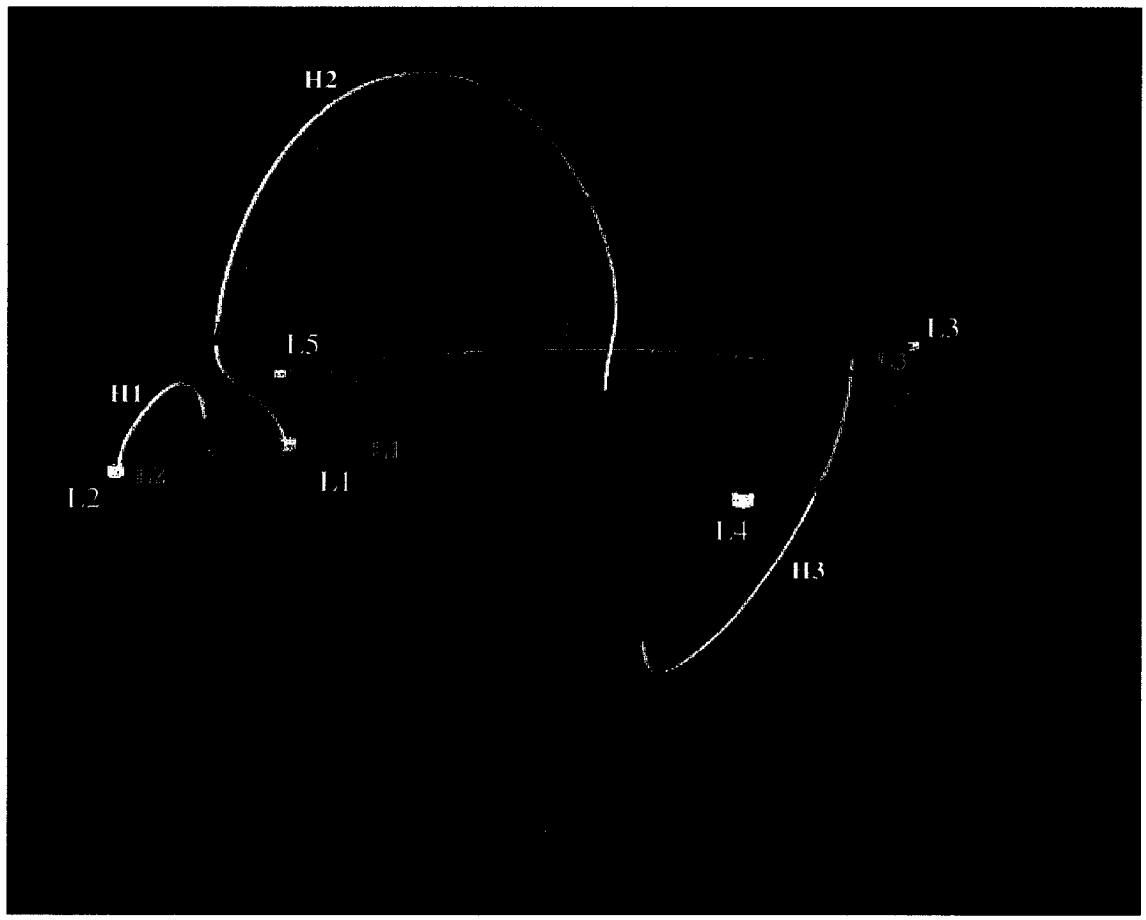


Figure 17: A bifurcation diagram for the Earth-Moon system

4.2.2 Annotation

Figure 17 shows a bifurcation diagram for the Earth-Moon System. All objects discussed in the above section are displayed. In this diagram, the following annotations are used.

Libration points

The Roman font is used to represent the libration points, both in the diagrams and in the text. For example, L1 represents the first libration point, *etc.*

Solution families and bifurcations

Each solution family is represented by a character followed by an integer. The character is a capital letter representing the family type, and the integer represents the libration point from which this family emanates, or to which this family is indirectly related.

The branch points along each family are represented by a character followed by two integers. The character, a capital letter, and the first integer, together represent the family to which the branch point belongs. The second integer represents the serial number of the branch point along from the family.

For example, the Lyapunov families are represented by the curves **L**1, **L**2, and **L**3 in Figure 17, where **L**i emanates from Li, respectively, $(i = 1, 2, 3)$. The *Vertical* families are represented by **V**i, where **V**i emanates from Li, respectively, with $i = 1, \cdots, 5$. The first libration point along **V**1, is represented by $V11$.

Table 1 gives a summary of the annotations.

Symbol	Definition
Li	Libration Point i, $(i = 1, \cdots, 5)$
Li	The planar Lyapunov family from Li, $(i = 1, 2, 3)$
Vi	The Vertical family from Li, $(i = 1, \cdots, 5)$
Si	The Short-Period planar Lyapunov family from Li $(i = 4, 5)$
Li	The Long-Period planar Lyapunov family from Li, $(i = 4, 5)$
Ai	The Axial family from $Li2$
Bi	The Back-flip family from $Vi2$
C1	The circular, planar family containing $V13$
C2	The circular, planar family containing $V23$, $V33$
Hi	The Halo family from $Li1$
Wi	The family from **V**i at $Vi1$, $(i = 4, 5)$
Lij	Bifurcation point j on **L**i
Vij	Bifurcation point j on **V**i
Hij	Bifurcation point j on **H**i, $(j = 1, 2)$

Table 1: Abbreviations.

4.3 The libration points of the Sun-Jupiter, the Earth-Moon, and the Sun-Earth system

In Chapter 2, an analytical method to compute the five libration points of the Earth-Moon system is discussed. In Chapter 3, we mentioned continuation to determine the dependence of the libration points on μ. The first method is used in PLAUT04 and the second method is used in the analysis of the dependence of the libration points on μ.

Table 2, 3, and 4 show the positions of the libration points in the rotating frame for the Sun-Jupiter, the Earth-Moon, and the Sun-Earth systems.

Libration Point	x	y	z
L1	0.93238638	0	0
L2	1.06880958	0	0
L3	-1.00039708	0	0
L4	0.49904700	0.866025404	0
L5	0.49904700	-0.866025404	0

Table 2: Positions of the libration points for the Sun-Jupiter system

Libration Point	x	y	z
L1	0.83691801	0	0
L2	1.15567991	0	0
L3	-1.00506240	0	0
L4	0.48785000	0.86602540	0
L5	0.48785000	-0.86602540	0

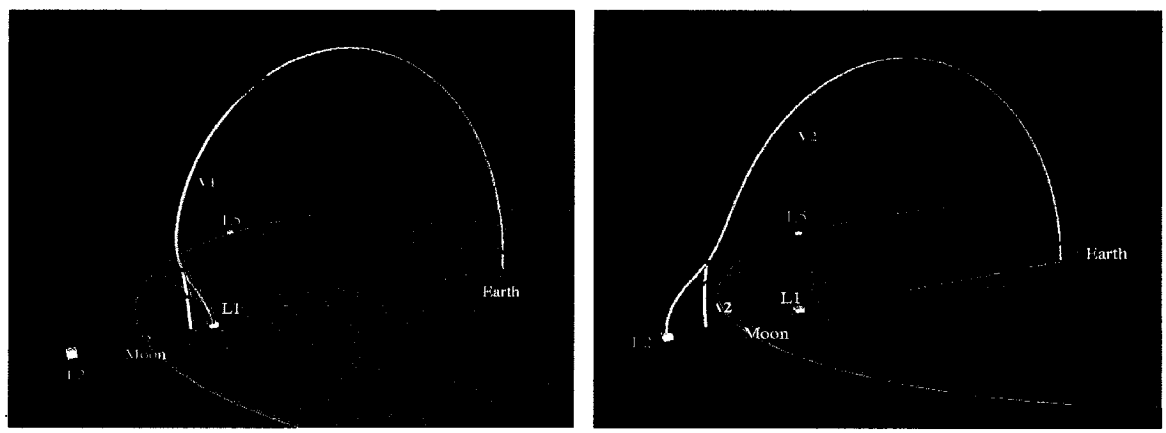
Table 3: Positions of the libration points for the Earth-Moon system

The dependence of the libration points on μ is shown in Figure 7. The diagram shows the libration points in the rotating coordinate system. Each of the libration points in the rotating coordinate system corresponds to a circle in the inertial system.

Given the libration points, we can set up the initial values for AUTO to compute the periodic families arising from them.

Libration Point	x	y	z
L1	0.99003044	0	0
L2	1.01003023	0	0
L3	-1.00000125	0	0
L4	0.49999700	0.866025404	0
L5	0.49999700	-0.866025404	0

Table 4: Positions of the libration points for the Sun-Earth system



(a) Bifurcation diagram around L1

(b) Bifurcation diagram around L2

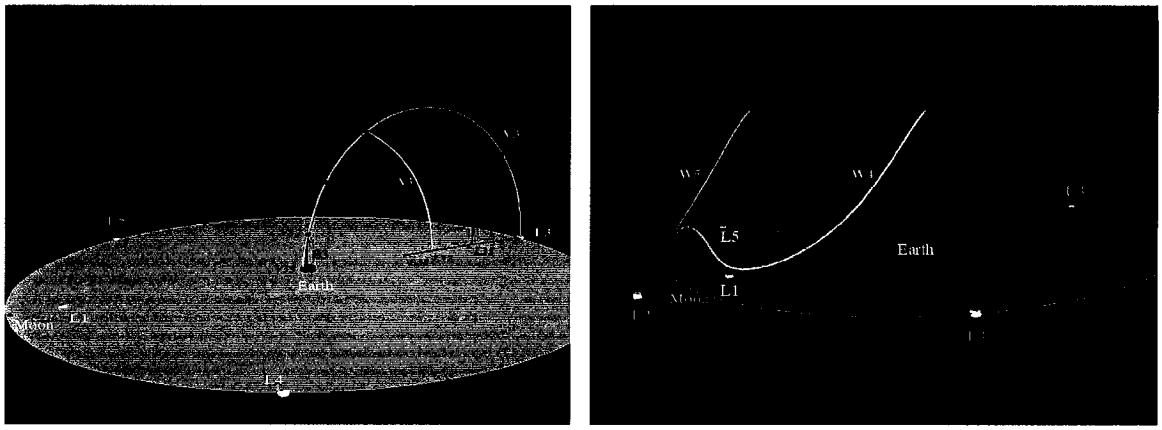
Figure 18: Bifurcation diagrams near the libration points for the Earth-Moon system

4.4 Computation of the bifurcation diagram for periodic solutions

Given the solutions for the libration points, starting from the libration points, and using AUTO, it is easy to compute the emanating families of periodic solutions. By following families that emanate from branch points along each *periodic family* (more accurately *family of periodic solutions*), a bifurcation diagram can be generated.

Figure 17 shows a bifurcation diagram for the Earth-Moon system. For the Sun-Earth, and the Sun-Jupiter system, similar bifurcation diagrams will be presented. Figure 18 gives closeup views of the bifurcation diagrams near the libration points of the Earth-Moon system.

Various families of periodic solutions are represented as curves in the bifurcation



(a) Bifurcation diagram around L3

(b) Bifurcation diagram around L4

Figure 18: Bifurcation diagrams near the libration points for the Earth-Moon system (cont'd)

diagram. Each point on a curve in the diagram represents a periodic solution.

Along families of periodic solutions, there are various branch points. It is important to note that in this thesis we reserve the term *branch point* (or *branching orbit*), for trans-critical and pitch-fork bifurcations, thereby excluding period-doubling, torus, and subharmonic bifurcations. However, the solution structure that we present can be viewed as a "skeleton", from which many other solutions may be reached [21, 30].

As we see, L1, L2, L3 lead to five solution families, $\mathbf{L}i$, $\mathbf{V}i$, $\mathbf{A}i$, $\mathbf{B}i$, and $\mathbf{H}i$, with $i = 1, 2, 3$. Because of the symmetry properties of the equations, L4 and L5 are symmetrical. For these two libration points, each leads to four solution families, which are named $\mathbf{L}i$, $\mathbf{S}i$, $\mathbf{V}i$, and $\mathbf{W}i$, respectively, with $i = 4, 5$. Although the branch points for L4 and L5 are symmetrical, we show both in the bifurcation diagram.

We also find that the Lyapunov family $\mathbf{L}i$, and the Vertical family $\mathbf{V}i$ emanate directly from Li, for $i = 1, \cdots, 5$. The *Short-period* planar Lyapunov family **S**4 and **S**5 also emanate directly from L4 and L5.

In Figure 17, the Lyapunov families are represented by the curves **L**1, **L**2, and **L**3, where $\mathbf{L}i$ emanates from the libration point Li ($i = 1, 2, 3$). There are two branch points along **L**1, called $L11$ and $L12$ for all three systems. There are three branch points, $L21$, $L22$, and $L23$, along **L**2 for the Earth-Moon and the Sun-Jupiter system, while we can only locate two, $L21$ and $L22$, for the Sun-Earth system. There are

three along **L**3, which are named $L31$, $L32$, and $L33$, for all the three systems.

Families bifurcating from **L**i ($i = 1, 2, 3$) at $Li1$ are called **H**i (the *Halo* families). Families bifurcating from **L**i ($i = 1, 2, 3$) at $Li2$ are called **A**i (the *Axial* families). Families bifurcating from **L**3 at $L33$ are **S**4/**S**5, which connect to L4/L5 at the other end. **L**1, **L**2, **L**3 themselves end in orbits that collide with the primaries.

Similarly, there are branch points along the Vertical families. Along **V**1, there are three such points, which are denoted as $V11$, $V12$, and $V13$. Along **V**2, there are also three branch points, namely, $V21$, $V22$, and $V23$.

Note, however, that there are four branch points along **V**3, namely, $V31$, $V32$, $V33$, and the branch point denoted by $V43$, where the **V**3 family connects to the family **V**4/**V**5. Along **V**4 and **V**5, there are $V41$ and $V51$, and the already mentioned branch point $V43$, which is located at the mid-point of **V**4/**V**5.

The Vertical families that emanate from L4 and L5 are smoothly connected. They can be considered as a single family, which we generally refer to as **V**4/**V**5.

In the bifurcation diagram, $L33$, $H11$, and $V43$ need more attention. Each of them connects to three different families. $L33$ connects **L**3 to **S**4/**S**5, which in turn connects L4 and L5. $H11$ connects **W**4/**W**5, and **H**1. $V43$ connects **V**4/**V**5, and **V**3. **C**2, a *Circular* family, is also a special family. From it, we can reach **H**1, **V**3, and **V**2.

In our AUTO computations, we found that the smaller μ, the more difficult it is to locate the bifurcation point $H11$ on **H**1. It is also difficult to reach **C**2 and **B**i ($i = 1, 2, 3$) for the Sun-Earth system. We have to increase the number of mesh points used, decrease the convergence requirements, and use very small step size (These can be controlled by the AUTO-constants NTST, EPSL, EPSU, DS, DSMAX, DSMIN; see [24]).

4.5 Stability of periodic solutions

For the systems we considered, most of the periodic orbits are unstable. Figure 19, 20, 21 show the stability of each family for the Earth-Moon, the Sun-Jupiter, and the Sun-Earth systems. From the diagrams we see that stable orbits appear along **S**4/**S**5, **V**4/**V**5, **W**4/**W**5, **L**4/**L**5, and **V**3. For both the Earth-Moon and the Sun-Jupiter systems, there always exist stable orbits along **B**2. For the Sun-Earth system, part

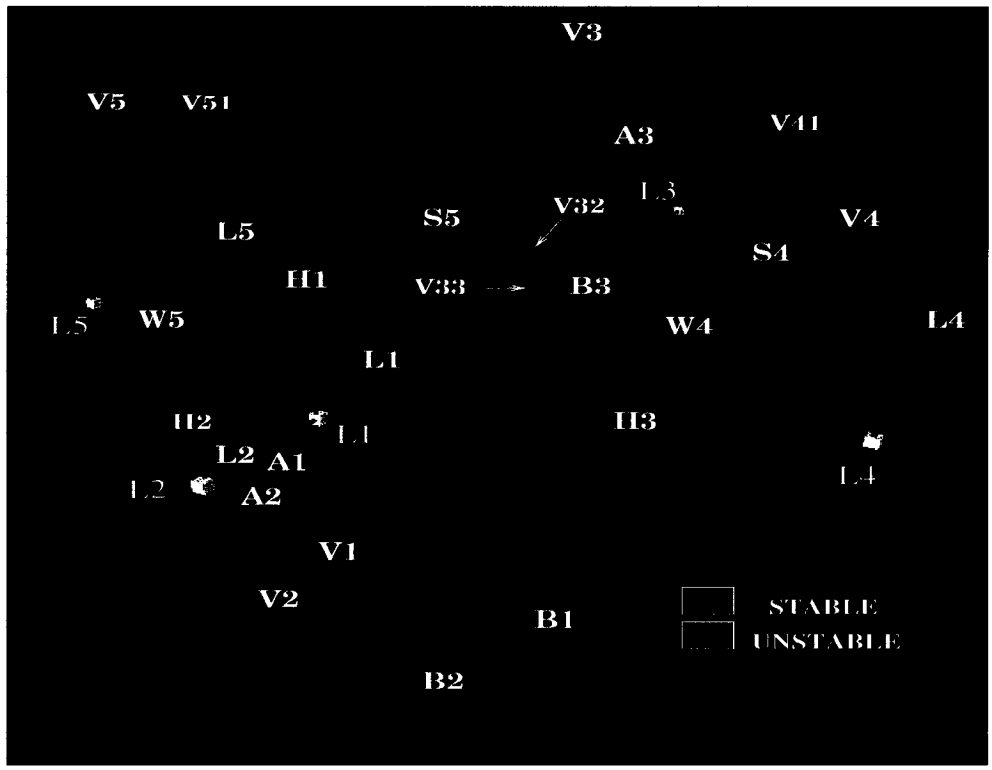


Figure 19: A bifurcation diagram with stability for the Earth-Moon system

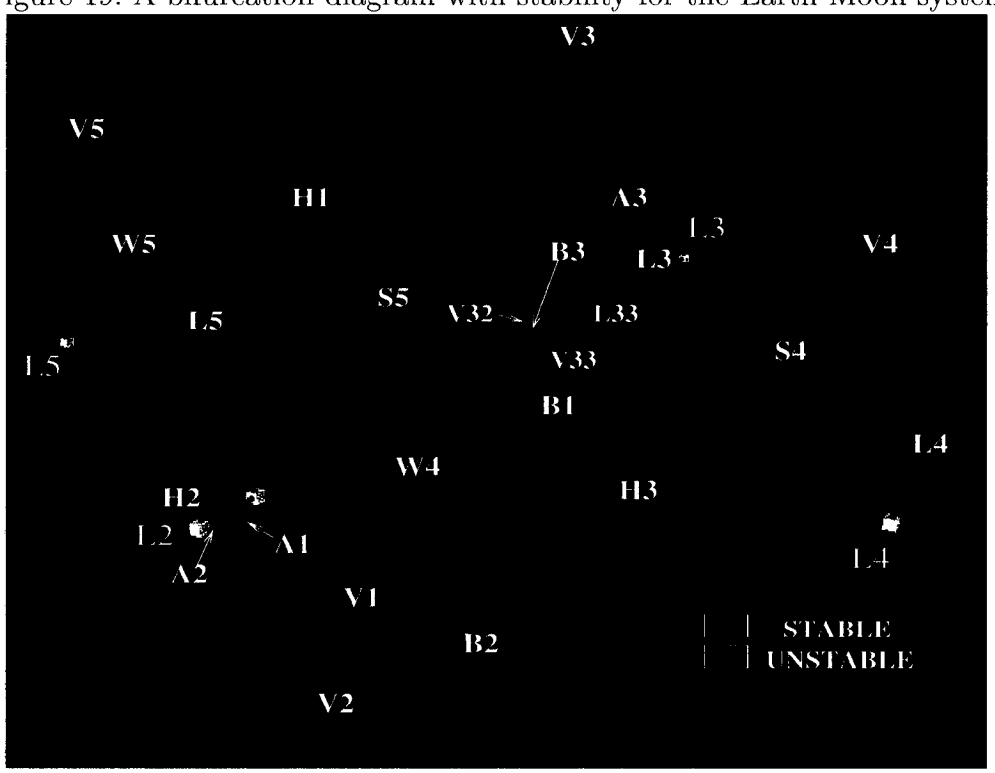


Figure 20: A bifurcation diagram with stability for the Sun-Jupiter system

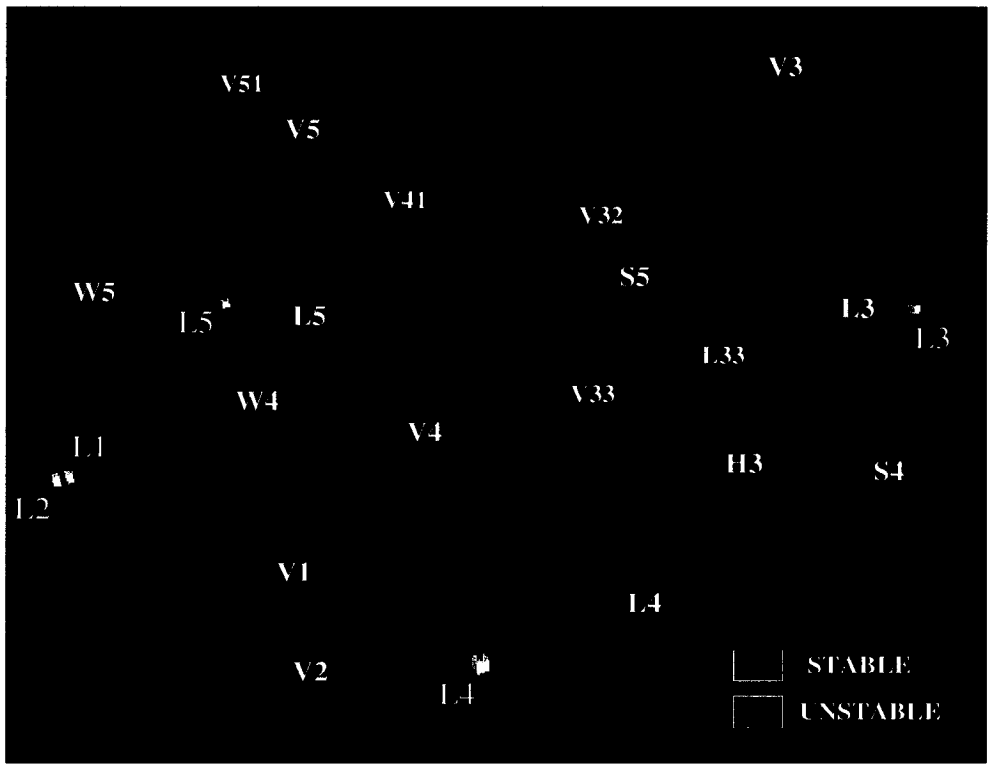


Figure 21: A bifurcation diagram with stability for the Sun-Earth system

of **W**4/**W**5 is also stable. Almost all orbits of the other families are unstable.

4.5.1 S4/S5

Figure 23 presents the change of the Floquet Multipliers for **S**4/**S**5 of the Sun-Jupiter system. Each subgraph of the Floquet Multipliers corresponds to a point in Figure 22. We take eight points from L4 to L5 along **S**4/**S**5, so we have eight pictures of their Floquet Multipliers. Figure 23(a) is related to point (1) in Figure 22. Figure 23(b) is related to point (2) in Figure 22, *etc.* Elsewhere in this chapter, the same method is used to show the Floquet Multipliers. From the diagram, we see that most **S**4/**S**5 orbits are stable. When we start from L4 , the orbits are stable until we reach point (2) on **S**4. All six Floquet Multipliers related to (1) lie on the unit circle, as shown in Figure 23(a). When we reach (2), the system changes from stable to unstable. The six Floquet Multipliers are very close to $(1, 0)$ (see Figure 23(b)). Beyond point (3), some of the six Floquet Multipliers are no longer on the unit circle. They split into three groups, where each group includes two points, The two points of the first group

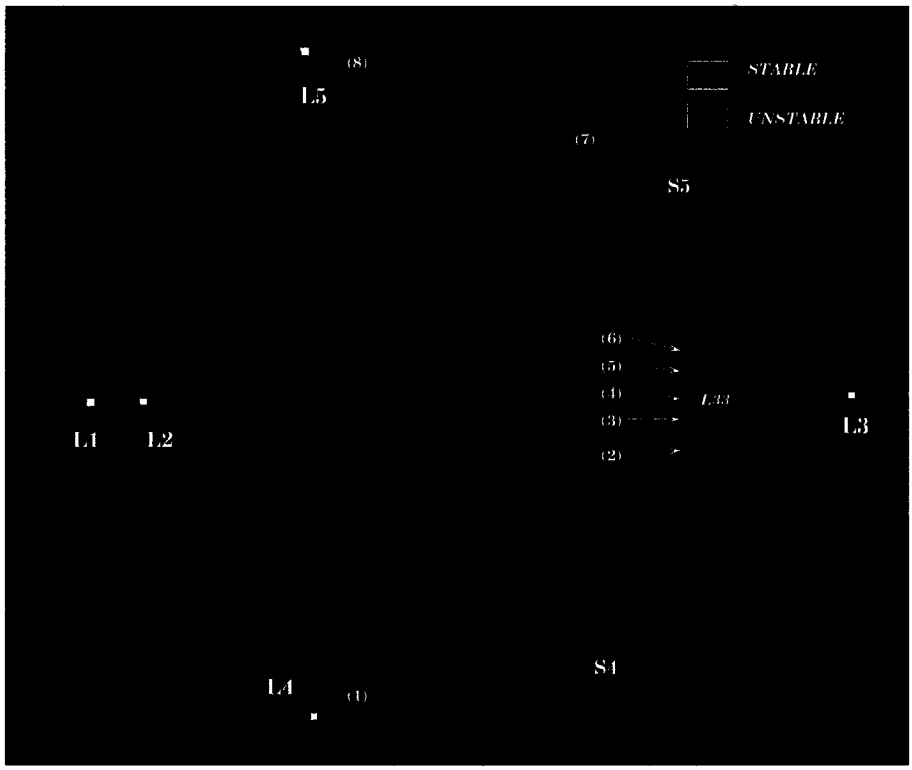


Figure 22: The family **S**4/**S**5 of the Sun-Jupiter system

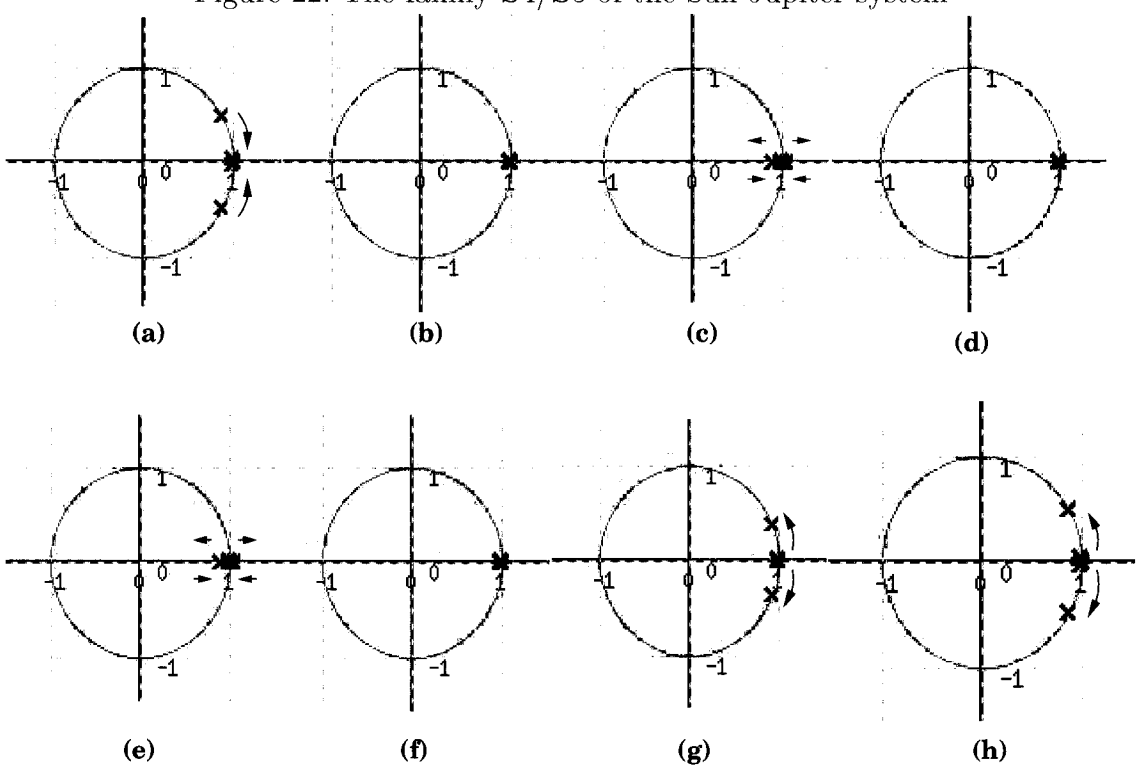


Figure 23: Floquet Multipliers along **S**4/**S**5 for the Sun-Jupiter system

are always at $(1,0)$. The other two groups first move away from $(1,0)$, then move back to $(1,0)$, along the x-axis but in the opposite direction as shown in Figure 23(c), 23(d). As we reach $L33$, the six Floquet Multipliers are very close to $(1,0)$ again. The change from point (8) to (4) is similar to that from point (1) to (4) (see Figure 23 (h), (g), (f), (e)).

The Earth-Moon and the Sun-Earth system have similar stability changes along their **S**4/**S**5 families.

4.5.2 V4/V5

There are also stable orbits along **V**4/**V**5. For the Sun-Earth system, the entire family **V**4/**V**5 is stable, and the stability extends to **W**4/**W**5. For the Earth-Moon system and the Sun-Jupiter system, only the parts from L4/L5 to its corresponding first branch point, namely, $V41/V51$, are stable. The remaining parts are unstable.

Figure 24 shows the family **V**4/**V**5 for the Earth-Moon system. The changes of the Floquet Multipliers are shown in Figure 25. The changes for the Sun-Jupiter system are similar to that of the Earth-Moon system. Figure 26 and 27 show the changes of the Floquet Multipliers for the family **V**4/**V**5 for the Sun-Earth system.

4.5.3 L4/L5

Stable orbits also occur along **L**4/**L**5. We have not found any regularity in the stability of these families.

4.5.4 V3

Stable orbits also occur along **V**3 for all three systems. The orbits between $V32$ and $V33$ are always stable. Figure 28 shows the stability of the **V**3 family of the Earth-Moon system.

4.5.5 B2

Along these families, the orbits are unstable when they bifurcate from **V**2. However, at some point they become stable, and later they change back to unstable again.

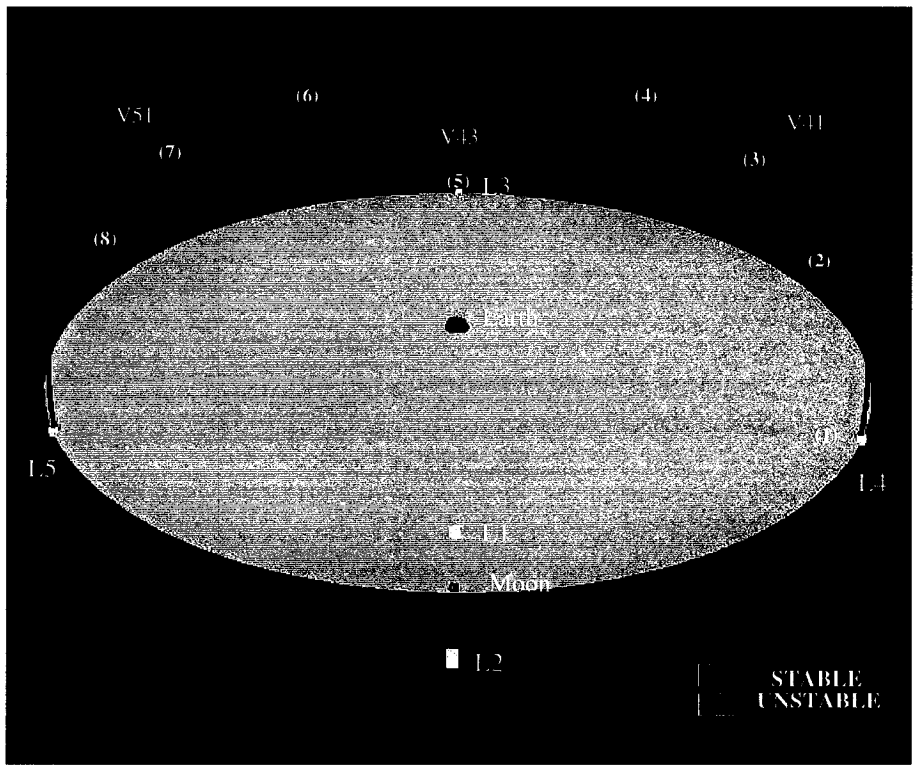


Figure 24: The family **V**4/**V**5 of the Earth-Moon system

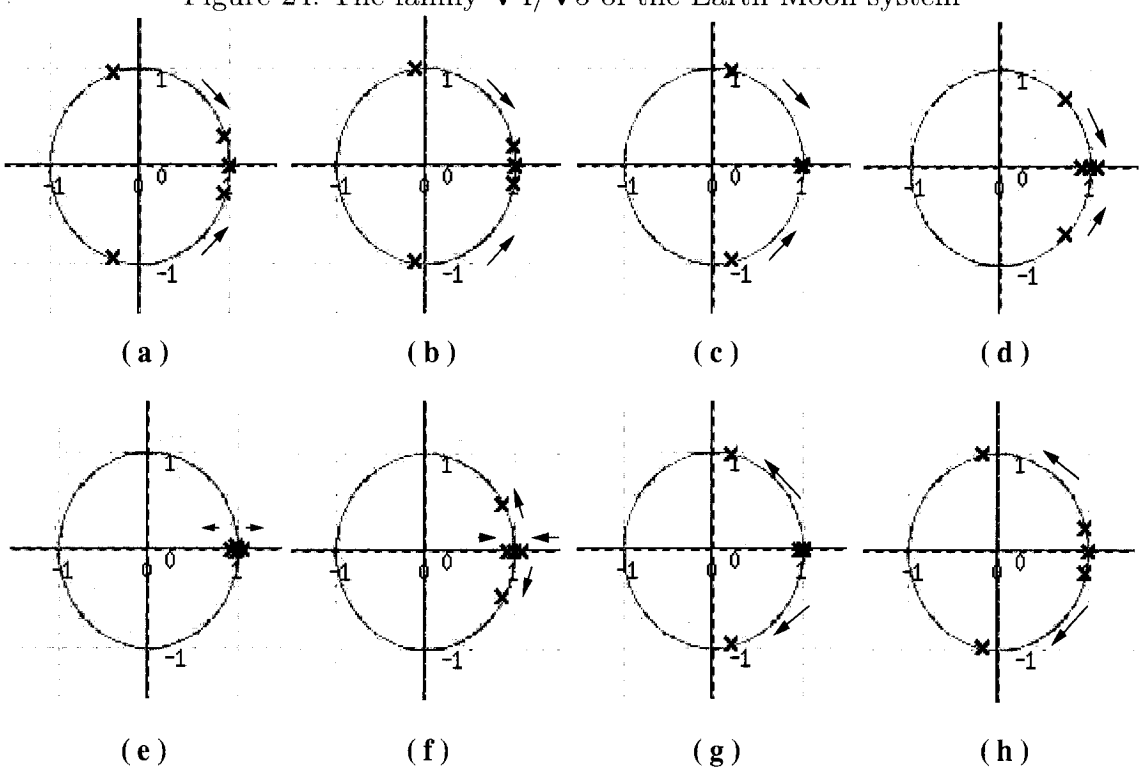


Figure 25: Floquet Multipliers along **V**4/**V**5 for the Earth-Moon system

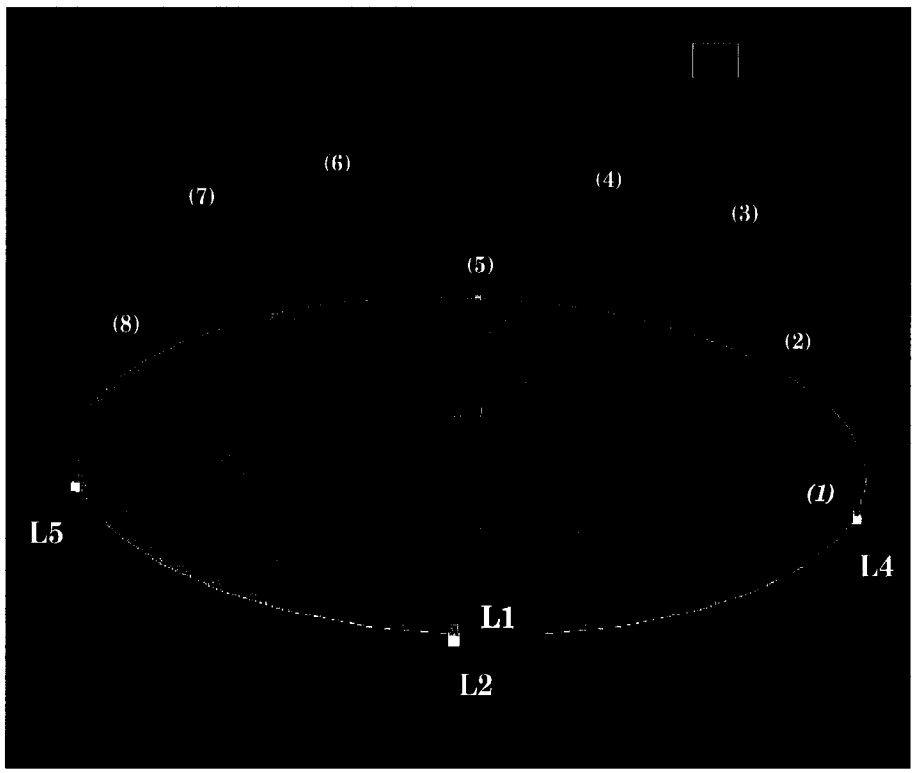


Figure 26: The family **V**4/**V**5 of the Sun-Earth system

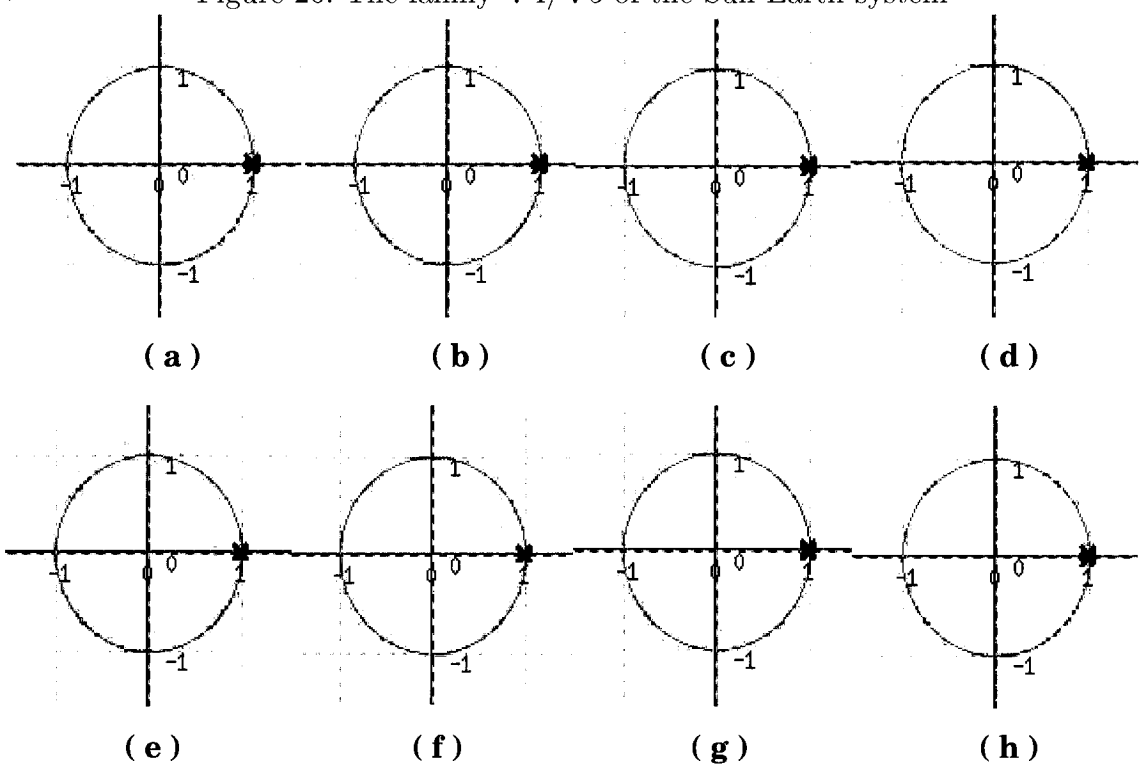


Figure 27: Floquet Multipliers along **V**4/**V**5 for the Sun-Earth system

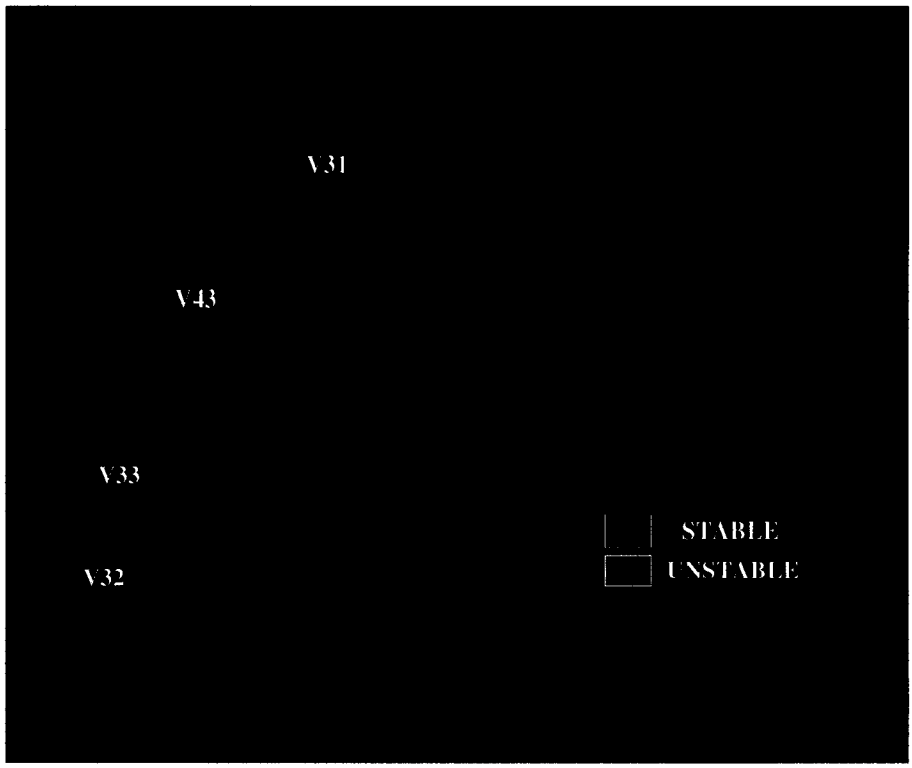
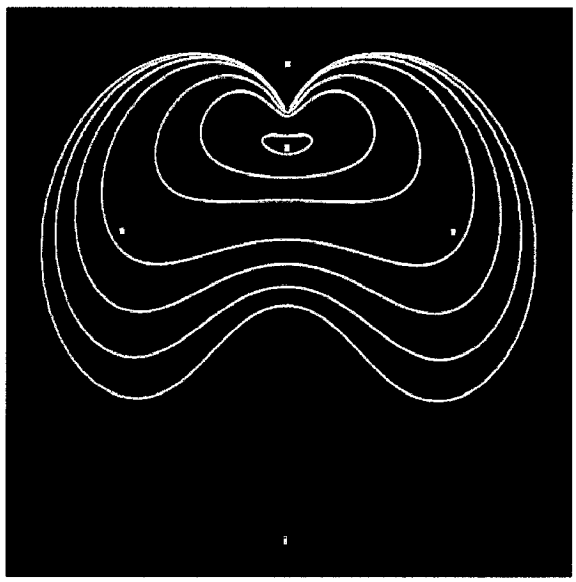


Figure 28: The **V**3 family of the Earth-Moon system

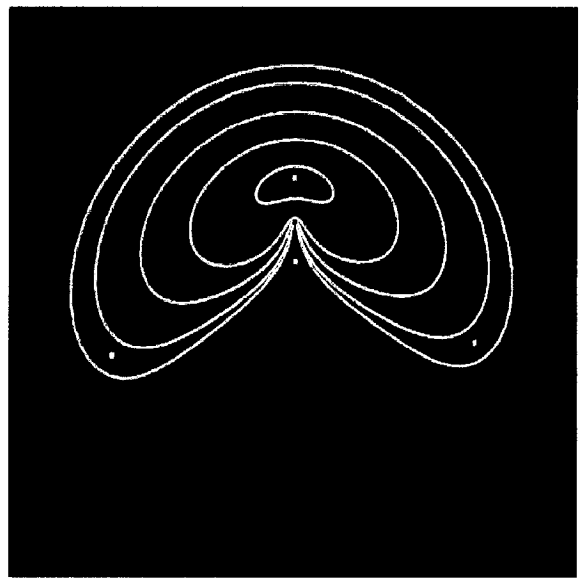
Both the **B**2 families of the Sun-Jupiter system and the Earth-Moon system have these characteristics.

4.6 The Lyapunov families

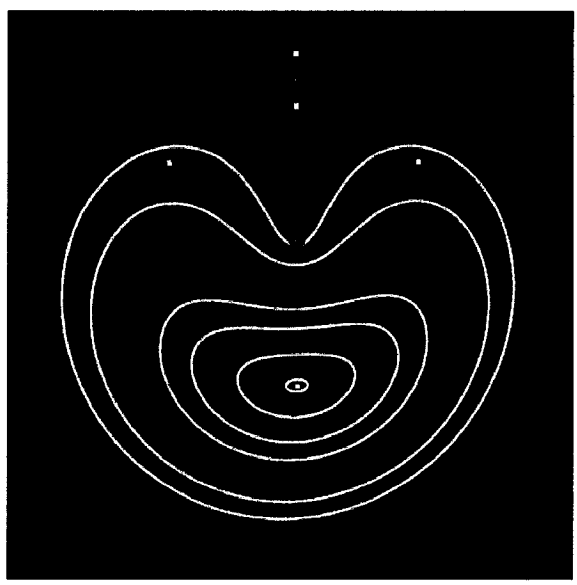
Figure 29 gives solution diagrams of Lyapunov families. In these diagrams, the branch point orbits are shown in a different color. The orbits of **L**1 are always located between the two primaries. With the increase of period, the orbit collides with the primaries. The **L**1 orbits approach both of the primaries as their period increases, and they finally collide with them. Orbits along **L**2 and **L**3 have different behavior from **L**1. The **L**2 orbits lie beyond the small primary, while the the **L**3 orbits lie beyond the large primary. With the increase of period, the orbits approach the primary. Finally the **L**2 orbits collide with the small primary, and the **L**3 orbits collide with the large primary.



(a) The **L**1 family

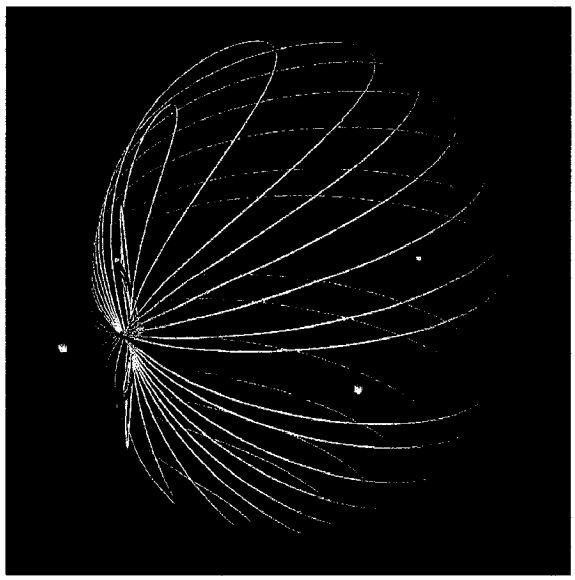


(b) The **L**2 family

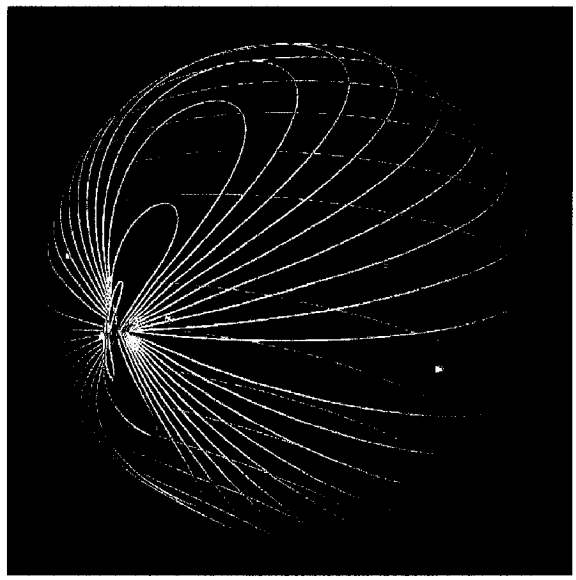


(c) The **L**3 family

Figure 29: Solutions along **L**i for the Earth-Moon system, $i = 1, 2, 3$

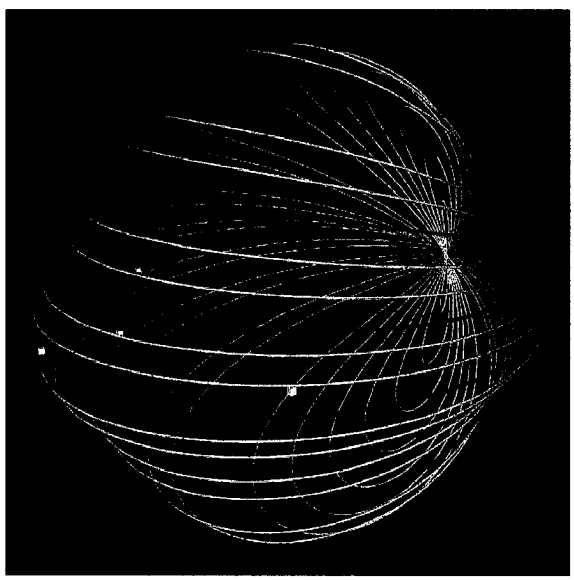


(a) The Vertical family **V**1

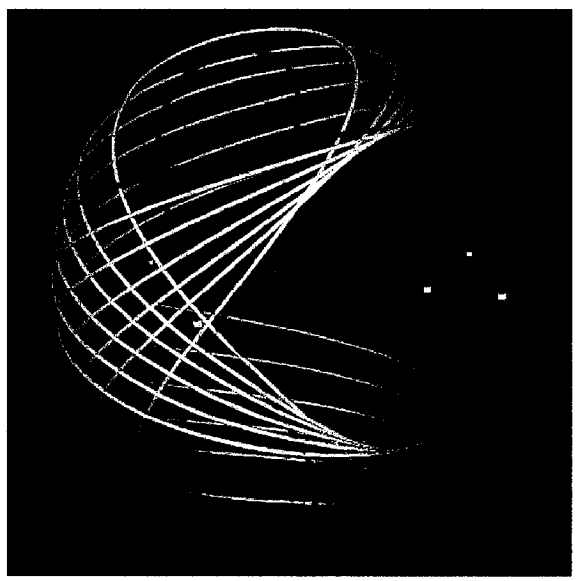


(b) The Vertical family **V**2

Figure 30: Solutions along **V**1 and **V**2 for the Earth-Moon system



(a) The Vertical family **V**3



(b) The Vertical family **V**4

Figure 31: Solutions along **V**3 and **V**4 for the Earth-Moon system

4.7 The Vertical families

Solution diagrams for the Vertical families **V**1 - **V**4 are shown in Figures 30, 31.

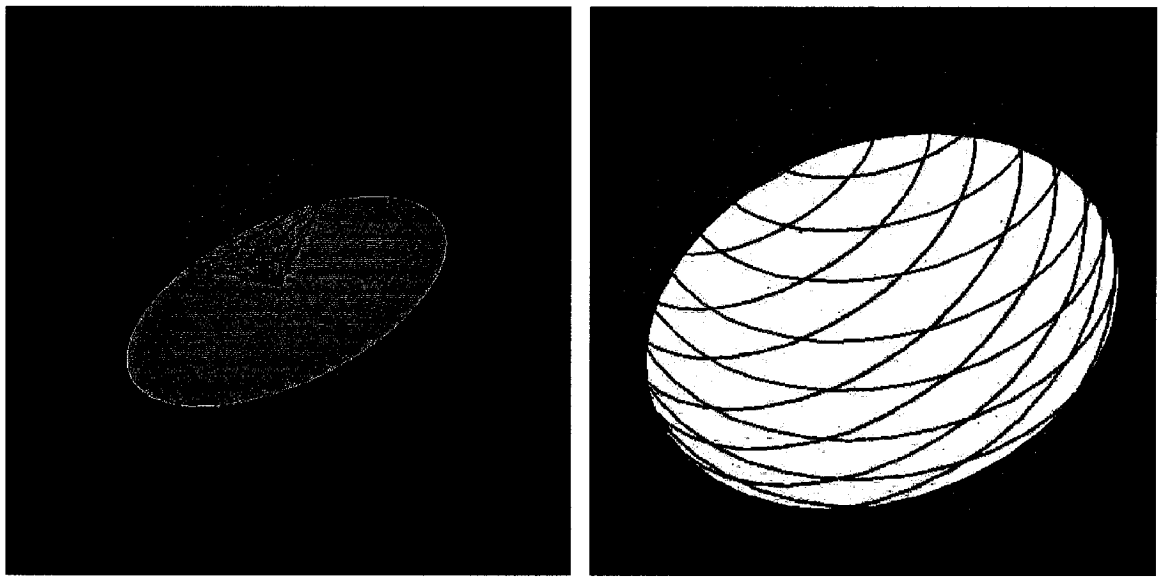
There are three branch points, $V11$, $V12$, and $V13$, along **V**1. There also exist three branch points, namely, $V21$, $V22$, and $V23$, along **V**2. However, there are four branch points, $V31$, $V32$, $V33$, and $V43$, along **V**3. These branch points are shown in Figure 17, and Figure 18. From $Vi1$, the Axial family **A**i bifurcates, for $i = 1, 2, 3$. The Back-flip family **B**i connects to the second branch point $Vi2$ ($i = 1, 2, 3$). The Circular planar family **C**i bifurcates from $Vi3$ ($i = 1, 2, 3$). $V43$ is the connection for the family **V**4/**V**5 and **V**3. Note that both $V23$ and $V33$ connect to the Circular planar family **C**2, which is also connected to **H**1 through $H12$. These branch points correspond to "reverse period-doubling bifurcations". For **V**1 and **V**2, the reverse period-doubling bifurcation corresponds to the third branch point along the family away from the libration point, whereas for **V**3, it corresponds to the fourth branch point. Nevertheless, the latter bifurcation will be referred to as $V33$.

The bifurcation orbits $V23$ and $V33$ encompass both primaries, whereas the bifurcation orbit $V13$ encompasses the large primary only. Furthermore, the orbits $V23$ and $V33$ belong to the same family of circular planar orbits, designated **C**2, whereas the planar orbit $V13$ belongs to another family of circular planar orbits, **C**1.

Figure 32(a) displays a vertical orbit of the Sun-Earth system, in the inertial frame centered at the small primary, in this case, the Earth. The orbit includes two parts. Half of it is above the x-y plane. The other half is below the x-y plane. The orbit generates two cone-like surfaces with the cone tops meeting near the small primary. Figure 32(b) displays the corresponding picture of this orbit in the inertial frame centered at the large primary, namely, the Sun.

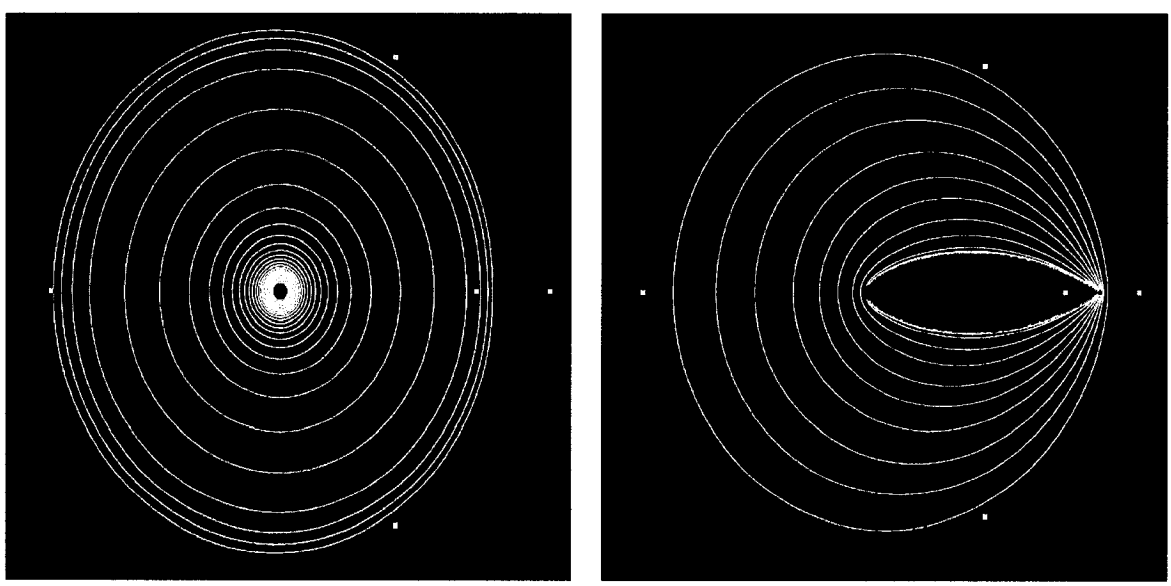
4.8 The Circular families

For the Earth-Moon system, the orbits along **C**1 encompass the Earth only. For decreasing period, they shrink toward the Earth. **C**2 encompasses both the Earth and the Moon. When the period of the **C**2 family decreases, orbits change from circles to ellipses, and finally collide with the primaries. **C**3 is different from **C**1 and **C**2. It includes two parts; the inner part only encompasses the Earth while the outer part encompasses both the Earth and the Moon. For increasing period, the diameter



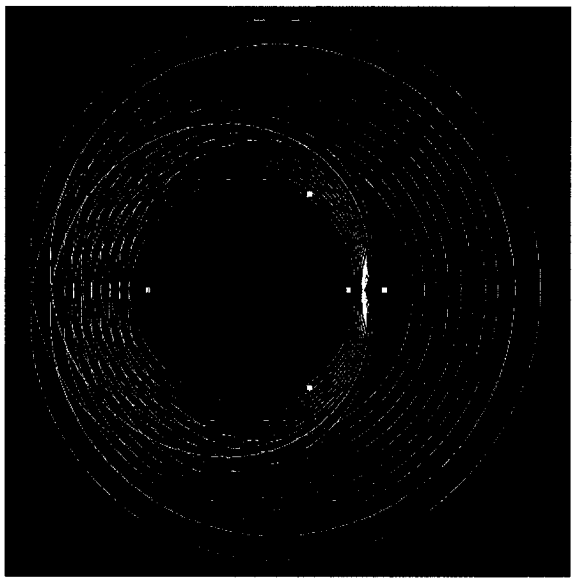
(a) Inertial frame (Earth-centered) (b) Inertial frame (Sun-centered)

Figure 32: Inertial frame view of a Sun-Earth **V**1 orbit



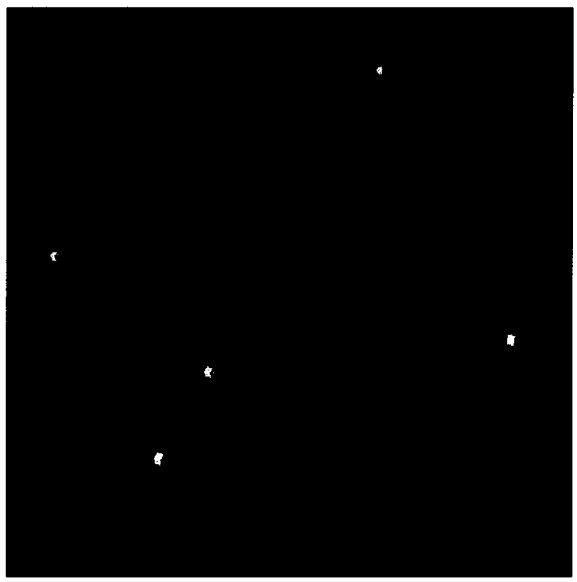
(a) The Circular family **C**1 (b) The Circular family **C**2

Figure 33: Solutions along **C**i for the Earth-Moon system, $i = 1, 2, 3$



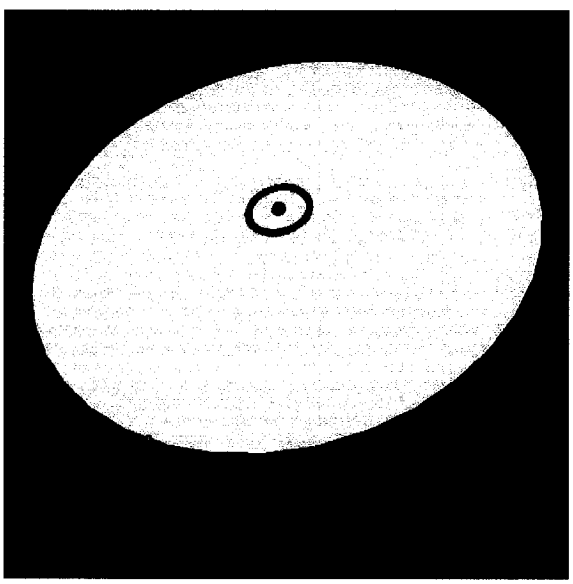
(c) The Circular family **C**3

Figure 33: Solutions along **C**i for the Earth-Moon system, $i = 1, 2, 3$ (cont'd)

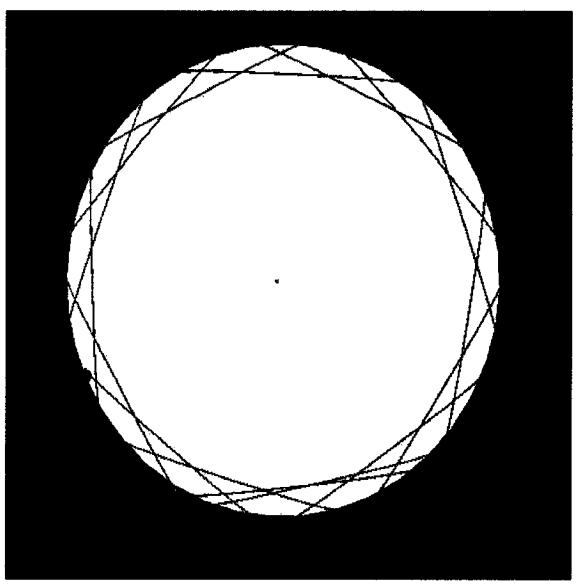


(a) Rotating frame

Figure 34: Different views of an Earth-Moon **C**1 orbit (I)

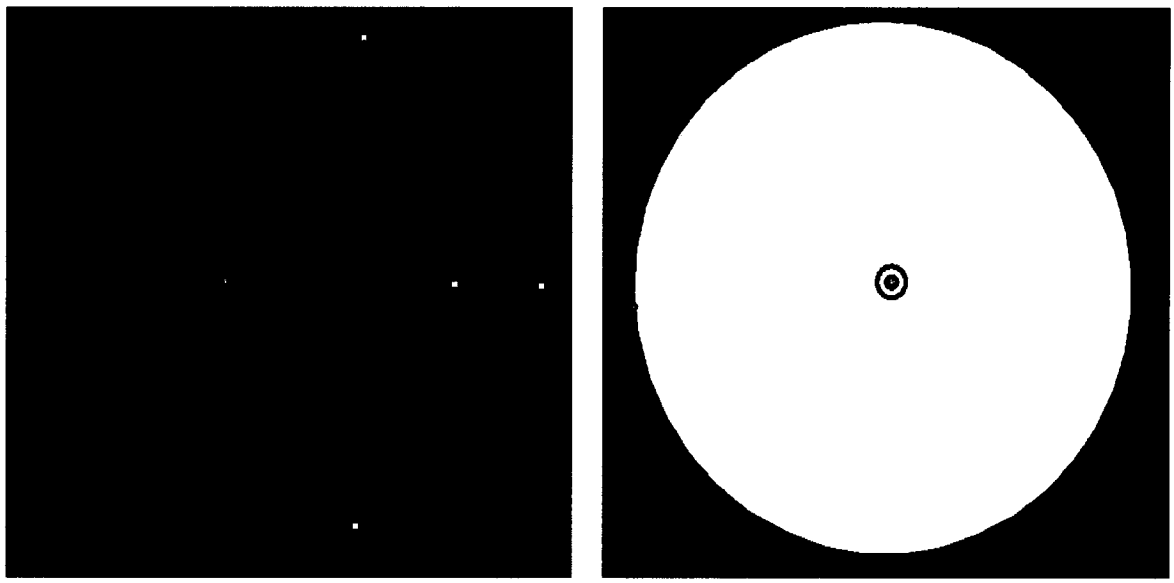


(b) Inertial frame (Earth-centered)



(e) Inertial frame (Moon-centered)

Figure 34: Different views of an Earth-Moon **C**1 orbit (I) (cont'd)



(a) Rotating frame (b) Inertial frame (Earth-centered)

Figure 35: Different views of an Earth-Moon **C**1 orbit (II)

of both parts increases, and finally the inner part approaches the Moon (see Figure 33).

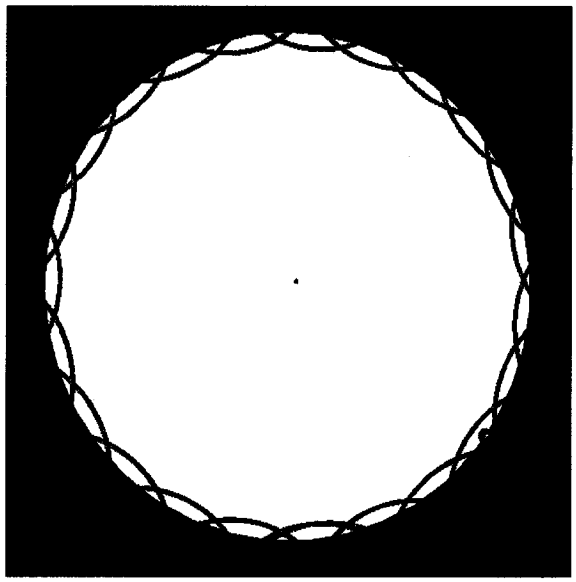
Some orbits of the family **C**i are shown in Figure 34, 35, 36.

4.9 The Halo families

Figure 17 shows three Halo families, namely, **H**1, **H**2, and **H**3. **H**1 starts from the second branch point along **L**1 and ends at **C**2. Along **H**1, there is one branch point, $H11$, which connects **H**1, **W**4, and **W**5. **H**2, and **H**3 also bifurcate from the corresponding Lyapunov families. The **H**2 orbits end in a collision with the small primary and the **H**3 orbits end in a collision with the large primary. At the collision, the orbits of **H**2 and **H**3 appear to become vertical to the x-y plane. The Halo families come in pairs, namely, the northern Halo families and the southern Halo families. The two halves are symmetrical according to the symmetry transformation $(x, y, z) \rightarrow (x, y, -z)$.

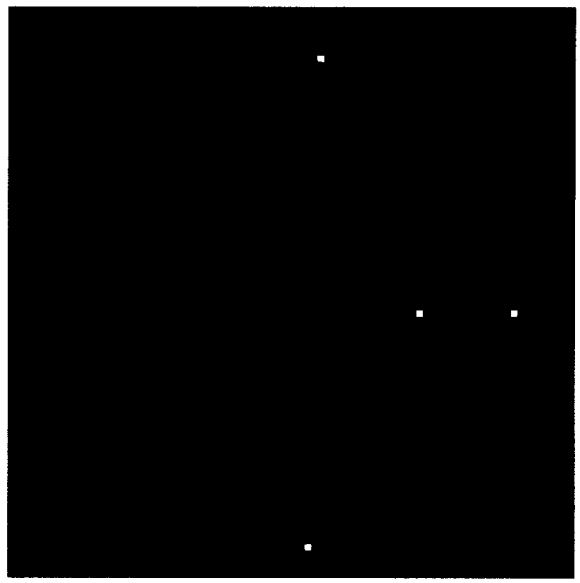
Solution diagrams of the northern Halo families are shown in Figure 37.

The first Halo orbit mission is the ISEE-3 (International Sun Earth Explorer 3)



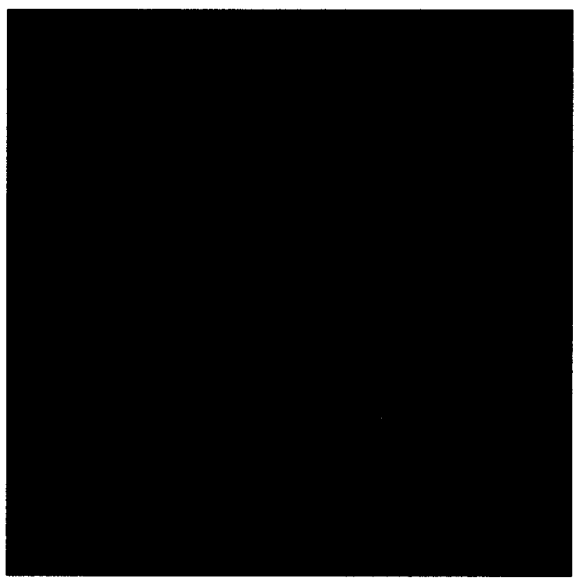
(c) Inertial frame (Moon-centered)

Figure 35: Different views of an Earth-Moon **C**1 orbit (II) (cont'd)

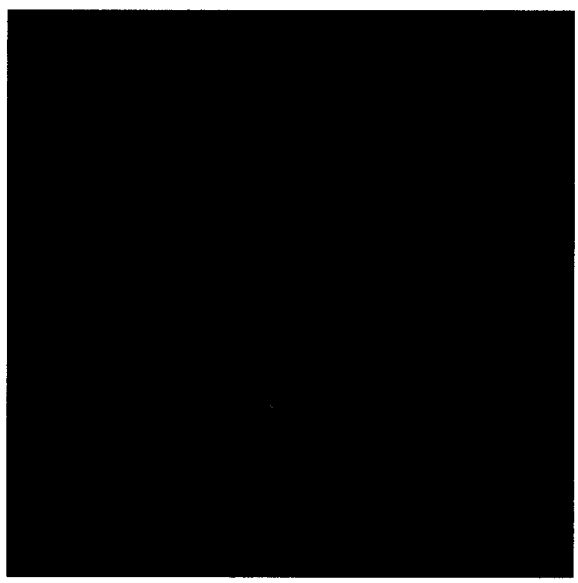


(a) Rotating frame

Figure 36: Different views of a **C**2 orbit of the Earth-Moon system

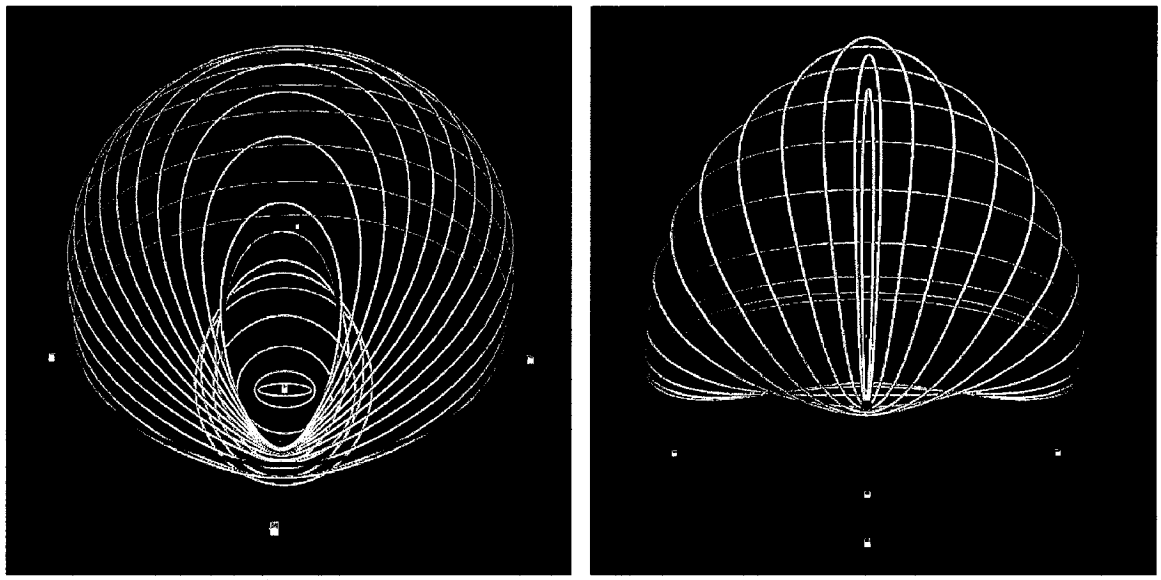


(b) Inertial frame (Earth-centered)



(e) Inertial frame (Moon-centered)

Figure 36: Different views of a **C**2 orbit of the Earth-Moon system (cont'd)



(a) The Halo family **H**1 (b) The Halo family **H**3

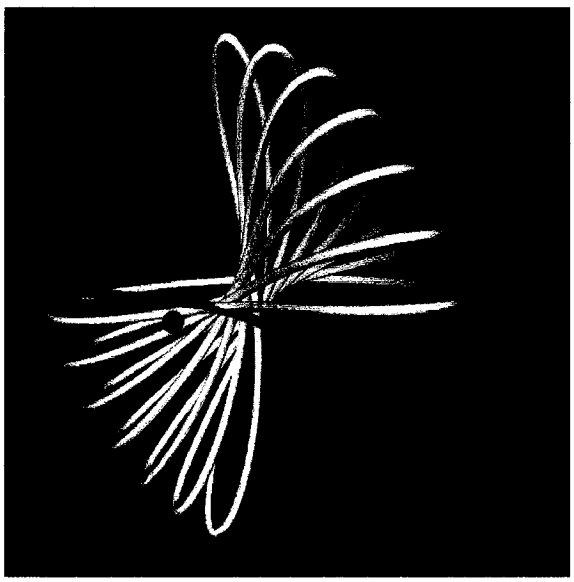
Figure 37: Solutions for **H**i of the Earth-Moon system

which was launched in 1978. The Solar Heliospheric Observatory (SOHO) was also in the Halo orbit. It was launched on December 2, 1995, and reached its final destination, the Sun-Earth L1 point, two months later, where it was injected into a Halo orbit, on February 14, 1996. The Genesis mission also followed a Halo orbit around the L1 libration point in the Sun-Earth system.

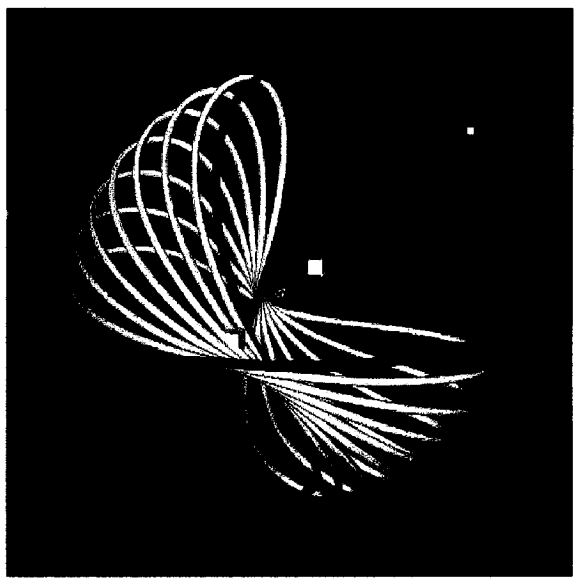
The term Halo derives from the appearance of the orbit that a spacecraft would seem to follow if it were viewed along the Earth-Sun line. Breakwell [12], Farquhar [31], and Howell [47] have studied the Halo orbits in detail.

4.10 The Axial families

The three Axial families are **A**1, **A**2, and **A**3. The name for these families come from the fact that each orbit is axially symmetric about the x-axis under the transformation y $\rightarrow$ -y, z $\rightarrow$ -z, t $\rightarrow$ -t. The entire family of orbits forms a loop. The Axial family **A**i connects to the corresponding Vertical family **V**i at one end, and to the Lyapunov family **L**i at the other end, respectively, with $i = 1, 2, 3$. Except for these two branch points, there are no further branch points along the Axial families. Figure 38 shows

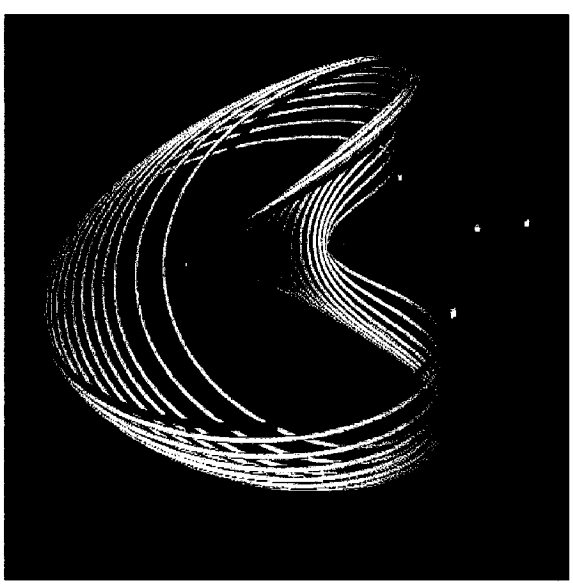


(a) The Axial family **A**1



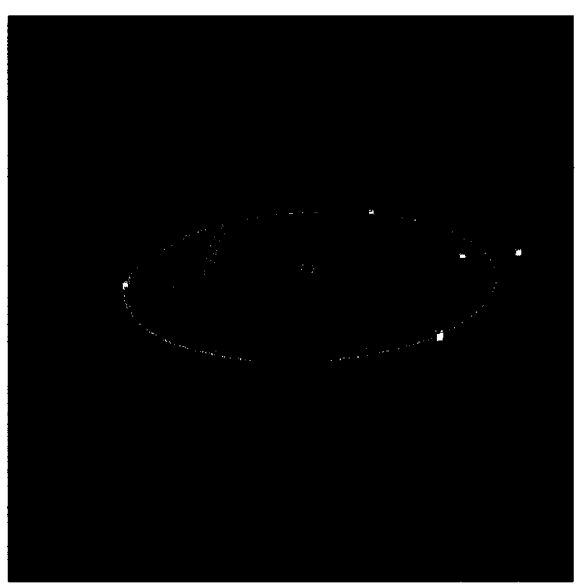
(b) The Axial family **A**2

Figure 38: Solutions for **A**i of the Earth-Moon system



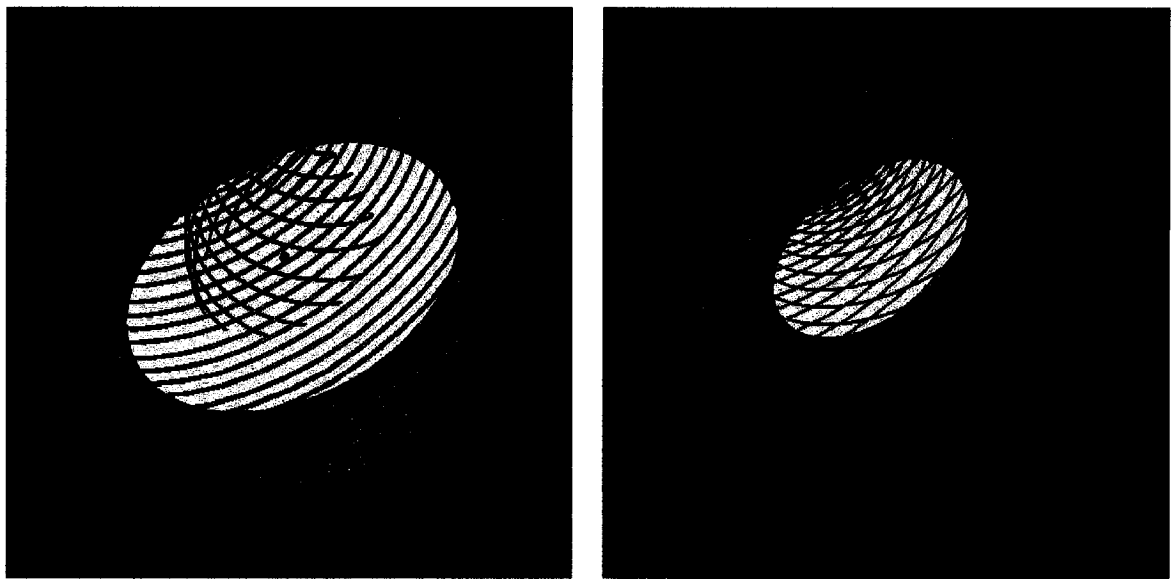
(c) The Axial family **A**3

Figure 38: Solutions for **A**i of the Earth-Moon system (cont'd)



(a) Rotating frame

Figure 39: Different views of an **A**3 orbit of the Earth-Moon system



(b) Inertial frame (Earth-centered) (c) Inertial frame (Moon-centered)

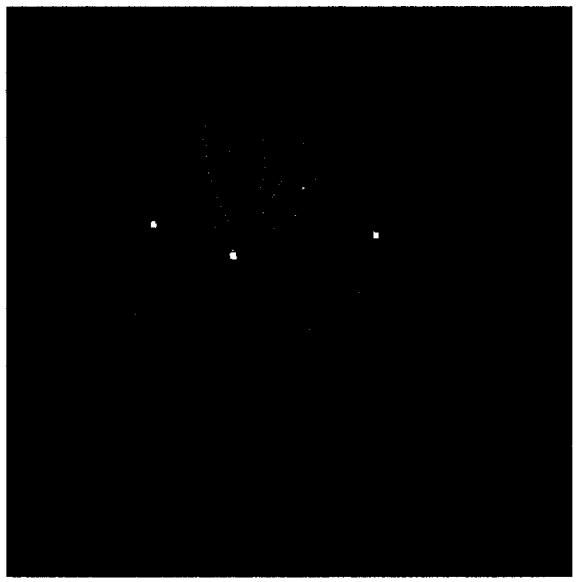
Figure 39: Different views of an **A**3 orbit of the Earth-Moon system (cont'd)

the solution diagrams for the Axial families of the Earth-Moon system.

Some of the orbits of the Axial family were plotted by Zagouras and Kazantzis for $\mu = 0.00095$ [88]. These orbits were also computed by Gómez and Mondelo [41] for the Earth-Moon system. An Axial orbit in the inertial system consists of two parts, the inner part (the blue part) and the outer part (the red part). In the Moon-centered picture (Figure 39(c)), the Moon is located at the geometric center of the picture, while the Earth is rotating around the Moon and its position is between of the inner and the outer parts of the infinitesimal.

4.11 The Back-flip families

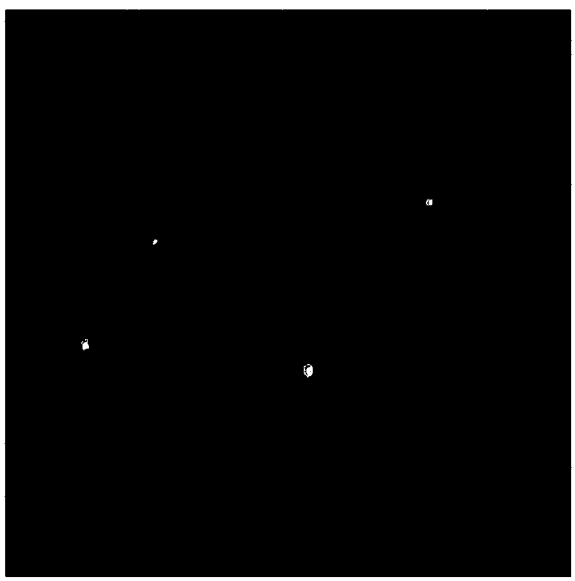
As mentioned above, the branch points $Vi2$ along the Vertical families $\mathbf{V}i$ lead to the Back-flip families. These families are named after the Back-flip maneuvers described in [85]. Each orbit of these families consists of two parts. Half of each orbit is above the x-y plane and half of each is below the x-y plane (see Figure 40). **B**1 ends at a collision with the small primary. **B**2 ends at a collision with the large primary. **B**3 does not end with a collision. It connects to another Circular planar family, namely,



(a) The Back-flip family **B**1

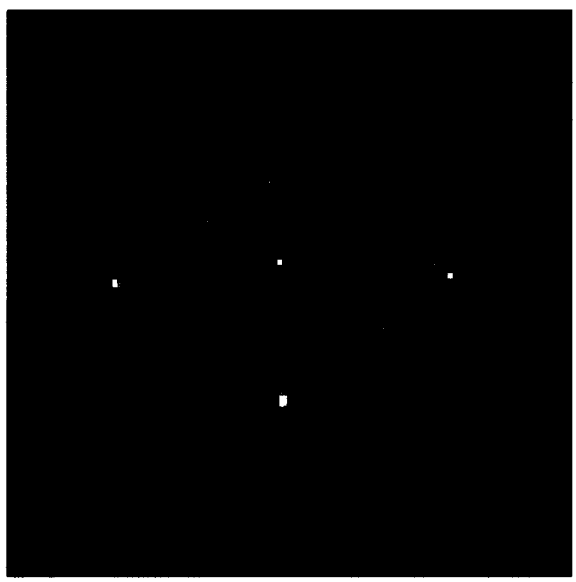
(b) The Back-flip family **B**2

Figure 40: Solutions for **B**i of the Earth-Moon system



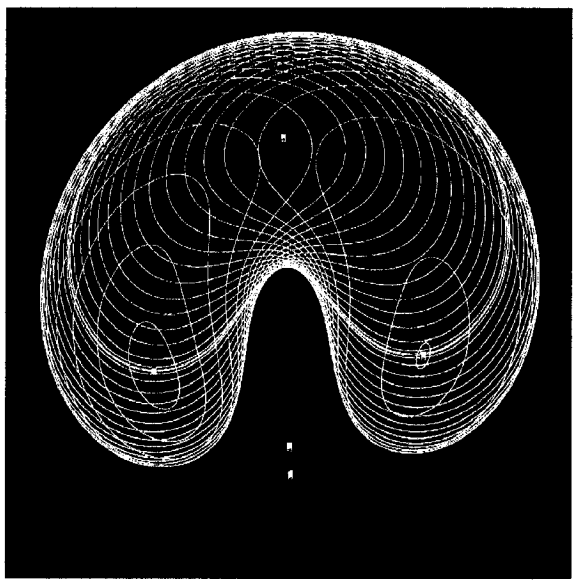
(c) The Back-flip family **B**3

Figure 40: Solutions for **B**i of the Earth-Moon system (cont'd)

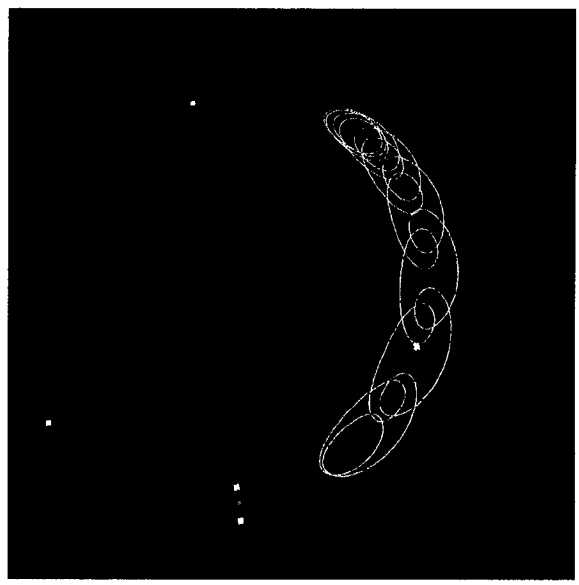


(a) The **W** family **W**4/**W**5

Figure 41: Solutions for **W**4, **S**4/**S**5, and **L**4/**L**5 of the Sun-Jupiter system



(b) The Short-period family **S**4/**S**5



(c) The Long-period family **L**4/**L**5

Figure 41: Solutions for **W**4, **S**4/**S**5, and **L**4/**L**5 of the Sun-Jupiter system (cont'd)

C3. Gómez and Mondelo [41] also computed a family of Back-flip orbits for the Earth-Moon system following a period-doubling bifurcation.

For decreasing mass ratio, the branch points $V12$, $V22$, approach **C**1, **C**2, along **V**1, **V**2, respectively. For very small mass ratios, the branch points $V12$ and $V22$ are very close. As a result it is difficult to switch branches at these points, in the case of the Sun-Earth system.

The Back-flip family **B**1 is very interesting and complicated. The orbits of this family can be divided into three groups based on their shape. D. J. Dichmann *et al.* [22] give a detailed description of these orbits.

4.12 The W, L and S families

W4 bifurcates from **V**4's first branch point, namely $V41$, and it connects to **W**5 and **H**1 at $H11$.

The families **S**4, **S**5, are planar families that bifurcate from **L**4, **L**5, respectively. Both **S**4/**S**5 connect to **L**3 at $L33$.

Another planar family bifurcates from L4, namely, the "Long-period" family **L**4,

while its symmetry partner **L**5 bifurcates from L5. The Long-period families reconnect to the Short-period families at the branch points $S41$ and $S51$. The nature of this reconnection depends on μ. For the Earth-Moon system, $S41$ and $S51$ correspond to period-quadrupling bifurcations along **S**4 and **S**5, respectively.

Figure 41 depicts solutions for **W**4, **L**4, and **L**5. The Long-period orbits have indeed "long periods". If the period of Jupiter rotating around the Sun is taken as one, even the shortest orbit in the diagram has a period over 78. The comutations also show that if the distance of an orbit from the libration point changes a little, its period increases greatly. In Figure 4.41(c), only four orbits are shown, the blue, green, yellow, and the red one. The blue one has the shortest period. The orbit becomes much more complicated when the period increases.

For decades most of the known periodic orbits were symmetrical. However, more and more asymmetrical periodic orbits are known today [14]. The Short-period and the Long-period families were the first known families of asymmetrical orbits These two families exist when the equilibrium point L4 is linearly stable, *i.e.*, when the mass ratio μ of the smallest primary to the total mass is smaller than 0.03852 [14].

In 1911, Brown [13] conjectured that the Long-period family has a final orbit doubly asymptotic to the unstable point L3; such a doubly asymptotic orbit is called a "*homoclinic orbit*". The numerical computation of J. Henrard [11] have shown that in the Sun-Jupiter case ($\mu = 0.000953875$), the Brown conjecture is not true.

4.13 Summary

In this chapter, AUTO was used to compute periodic solutions of the CR3BP. Results for the Earth-Moon, the Sun-Jupiter, and the Sun-Earth systems were given. Bifurcation diagrams of these systems were also constructed from these solutions. The stability of these solutions along some families was also discussed.

Chapter 5

Development of Data Visualization and Animation Software

5.1 Introduction

Graphical viewing of scientific data helps to understand them. Data visualization for scientific calculations has been an important area in Computer Science, and it is foreseeable that this trend will continue in the future. The CR3BP is a very complicated problem. The datasets that contain its solutions can be large. For example, for one orbit family, we may have several Megabytes of data. It is almost impossible to read these data. A good graphics tool is necessary in this research. As one of the best tools for ODE bifurcation analysis, AUTO has an old graphics tool, which was developed in the time of black-and-white graphics. Its drawbacks include:

- it does not have good 3D graphics capability;
- it cannot animate solutions;
- the command line interface is an obstacle for new users.

These factors motivated us to develop a new tool. In this chapter, the development of the new graphics and animation tool for visualization of AUTO data is discussed.

5.2 Objectives

The new tool, called PLAUT04, focuses on the following objectives:

1. *Usability.* A disadvantage of the old AUTO graphics tool is that it is hard to learn and hard to use. The old graphics tool uses command-driven techniques. This results in a steep learning curve for new users. Users have to memorize the commands and know how to use them. The more powerful computers and user-friendly graphical OS are used, the less command line interfaces are used. The graphical interface and event-driven techniques in PLAUT04 provide an easy-to-learn, easy-to-use, user-friendly environment.

2. *Generality.* PLAUT04 can be used for general AUTO data. An extended version of PLAUT04, called PLAUT04/r3b, is developed especially for data visualization of the CR3BP.

3. *Flexibility.* The old graphical tool has limited 3D capabilities. This requires a user to have a powerful imagination to reconstruct 3D pictures of a system mentally. Although, this is not a problem for some users, it is difficult for most people. Allowing users to view data in a 3D environment is helpful.

4. *Animation.* The capability of animation of solutions is very useful for understanding them. PLAUT04 provides not only general purpose animation, but also extended functions for the CR3BP. These extended functions include, (i) showing the data in both the rotating and the inertial frames. (ii) selecting the origin of the inertial frame.

In the implementation, aspects of human-computer interface and software engineering are also considered.

5.3 Development environment and architecture

AUTO was originally written in *FORTRAN*. However, AUTO2000 has provided a command line interface based on Python, and the source codes have been rewritten in C. AUTO is mainly used under Linux, so we choose Linux as our development platform.

For better portability to different Unix or Unix-like platforms, Motif is selected as our GUI programming library. Open Inventor is chosen to create the 3D interactive graphics, and C/C++ is chosen as the programming language for PLAUT04. Figure 42 shows the architecture of PLAUT04.

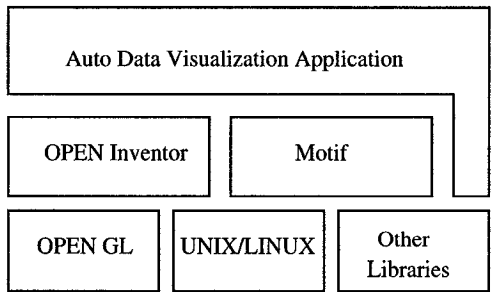


Figure 42: Architecture of PLAUT04

5.4 Overview of Open Inventor

According to SGI, Open Inventor is "an object-oriented 3D toolkit offering a comprehensive solution to interactive graphics programming problems. It presents a programming model based on a 3D scene database that dramatically simplifies graphics programming. It includes a rich set of objects such as cubes, polygons, text, materials, cameras, lights, trackballs, handle boxes, 3D viewers, and editors that speed up your programming time and extend your 3D programming capabilities" [81].

Open Inventor has been widely used in 3D graphical visualization, and has become the *de facto* standard 3D graphical API for complex scientific and engineering 3D visualization, and visual simulation applications.

Based on OpenGL, Open Inventor objects include database primitives, such as shape, property, group, and engine objects; interactive manipulators, such as the handle box and trackball; and components, such as the material editor, directional light editor, and examiner viewer.

There are three different implementations of the Open Inventor API.

1. The first and the most popular implementation is developed by SGI. Initially, the use of this implementation was based on a SGI license. On August 15, 2000, SGI released the source codes to the open source community. The latest version of this implementation is 2.1.5.10 as of August, 2004.

2. The second widely used open source implementation of the Open Inventor API is called *Coin*. The Coin Application Programmer's Interface (API) is fully compatible with SGI's Open Inventor v2.1, and it also incorporates many new features.

Coin is sponsored by Systems In Motion. Coin 2.2 was released on January 15, 2004.

3. The third implementation is from TGS. This implementation is a commercial package.

These three implementations are source code compatible. Coin is used for the development of PLAUT04,

5.5 User requirement specification

5.5.1 User requirements

- Starting the program.

  A user starts the program from the command line.

  A user should be able to read both datasets in the current directory and datasets in other directories.

  A resource file is needed for a user to set up predefined parameters.

- Selecting datasets to be drawn (solution/bifurcation).

  After the program is started, a user should be able to select any combination of the columns in AUTO data files to be shown. A user should be able to select the type of diagram to be drawn — either the solution diagram or the bifurcation diagram.

- Selecting graphics type (curve, tube, surface).

  The GUI should also provide different styles, such as curves, tubes, or surface(s), for the graphics.

- Adding/ removing graph widgets (coordinate axes, primaries, reference plane, libration points).

  The widgets, such as the coordinate axes, primaries, reference plane, and libration points, may be useful. The capability of adding widgets to/removing them from the scene graph is also important.

- Solution and orbit animation.

  Solution and orbit animation help to understand the orbit changes, and help to predict the tendencies of these changes.

- Changing frames of the system (inertial frame, rotating frame).

  For the CR3BP, the model is built in a rotating frame. However, graphics in an inertial frame is more intuitive. The capability of converting the graphics from the rotating frame to the inertial frame, or reversely, is useful for the CR3BP.

- Picking data.

  A way of showing the data corresponding to an interesting point in the graphics is also required.

- Setting default values.

  A user's current settings should be saved in a default resource file, so that the same settings can be used in the future.

- Manipulating the graphics (zoom in/zoom out/rotate).

  A user should be able to zoom in and zoom out, in order to see the details of an interesting part. A user should be able to view the graph from different directions.

- Exporting the graphics to an Open Inventor text file.

  Saving the graphics in an Open Inventor text file is useful. This allows users, who do not have PLAUT04 installed on their system, to view the graphics generated from PLAUT04 , when they have Open Inventor installed.

- Stopping the program

  A user must be able to stop the program at any time.

5.5.2 Non-functional requirements

Hardware

CPU: Pentium III 800 or higher.

Memory: 128 Megabytes or higher.

Other:

At least 1 Megabyte hard drive space available for the software installation.

Software

Operating System: Red Hat Linux 7.3 or higher.

Applications:

OpenGL or Mesa is essential;
SGI Open Inventor 2.1.5.10, or Coin3D 2.2 with SoXt 1.1.0 and simage 1.6 must be included in the system;
Motif 2.0 or later is required.

5.6 System design and implementation

5.6.1 The system architecture

System components

Figure 43 depicts the system components. It contains four components: GUI, data parser, data model (or data object), and graphical object.

The GUI component is responsible for querying the data object component, where all the persistent data is stored using a linked list. All the data in the data object is parsed from the AUTO data files, "*s.xxx*", "*b.xxx*", and "*d.xxx*", by the data parser component. The AUTO "s.xxx" file stores information about solutions; the "b.xxx" file stores information about the bifurcation diagram; and the "d.xxx" file stores Floquet Multipliers and other diagnostic information. The graphical object component gets data from the data object and generates the diagram.

The GUI component allows users to achieve the complete functionality, and is implemented using widgets provided by Motif. The GUI component constructs the entire user interface.

Since the other components are independent of each other, the GUI component

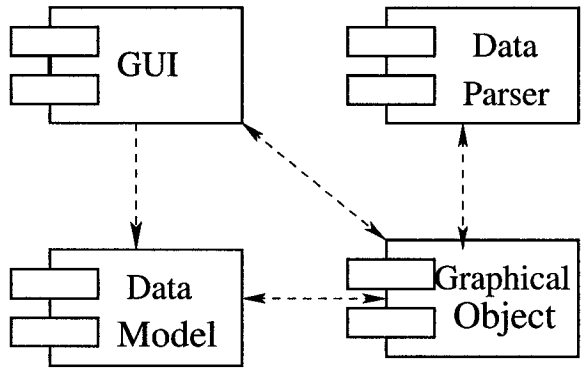


Figure 43: System component diagram

has to bridge the gaps between other components. It is the actual "control center" of PLAUT04. It is responsible for manipulating the scene and controlling the system. From the GUI, a user can give commands to change graphics type and style, add/remove widgets, and animate solutions, *etc.*

The graphical component maps the data from the dataset to the screen in colored graphics based on user's choice.

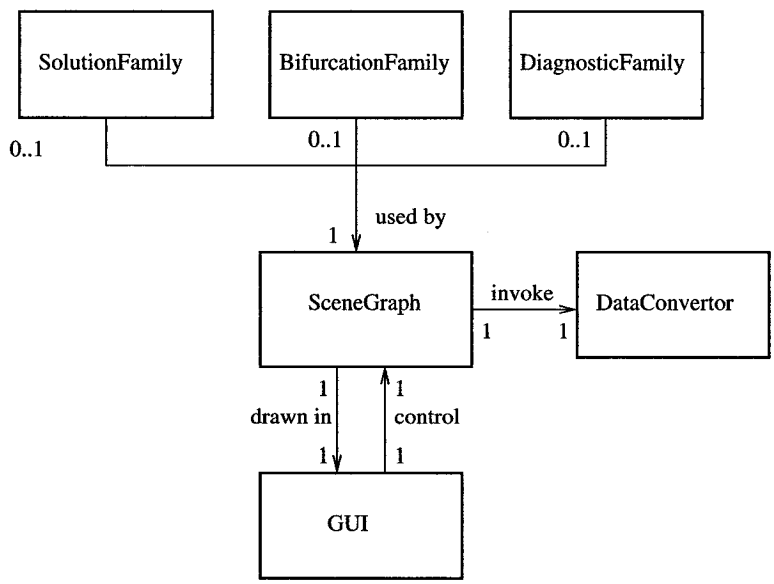


Figure 44: System class diagram

Figure 44 shows the classes that comprise the package. Based on their functionality, the classes can be divided into three main groups: classes related to the GUI, classes related to the AUTO data parsing and manipulation, and classes related to the scene graph generation.

The GUI class is comprised of various Motif Widgets, such as `XmButton`, `XmFrame`, `XmPanel`, `XmLabel`, `XmDialog`, `XmCombobox`, `XmList`, *etc.* Due to their detailed nature, they are not modeled in the diagram.

The `SolutionFamily` class, the `BifurcationFamily` class, and the `DiagnosticFamily` class focus on AUTO data parsing and AUTO data manipulation. The database for the system is also created and maintained with these classes.

The `DataConvertor` class is a utility class which converts the AUTO data from a rotating system to an inertial system.

The `SceneGraph` class creates the scene graph. It reads AUTO solution and bifurcation data from the database created by the AUTO data manipulation classes, and it also uses the system environment variables which are set by the GUI component.

System state diagram

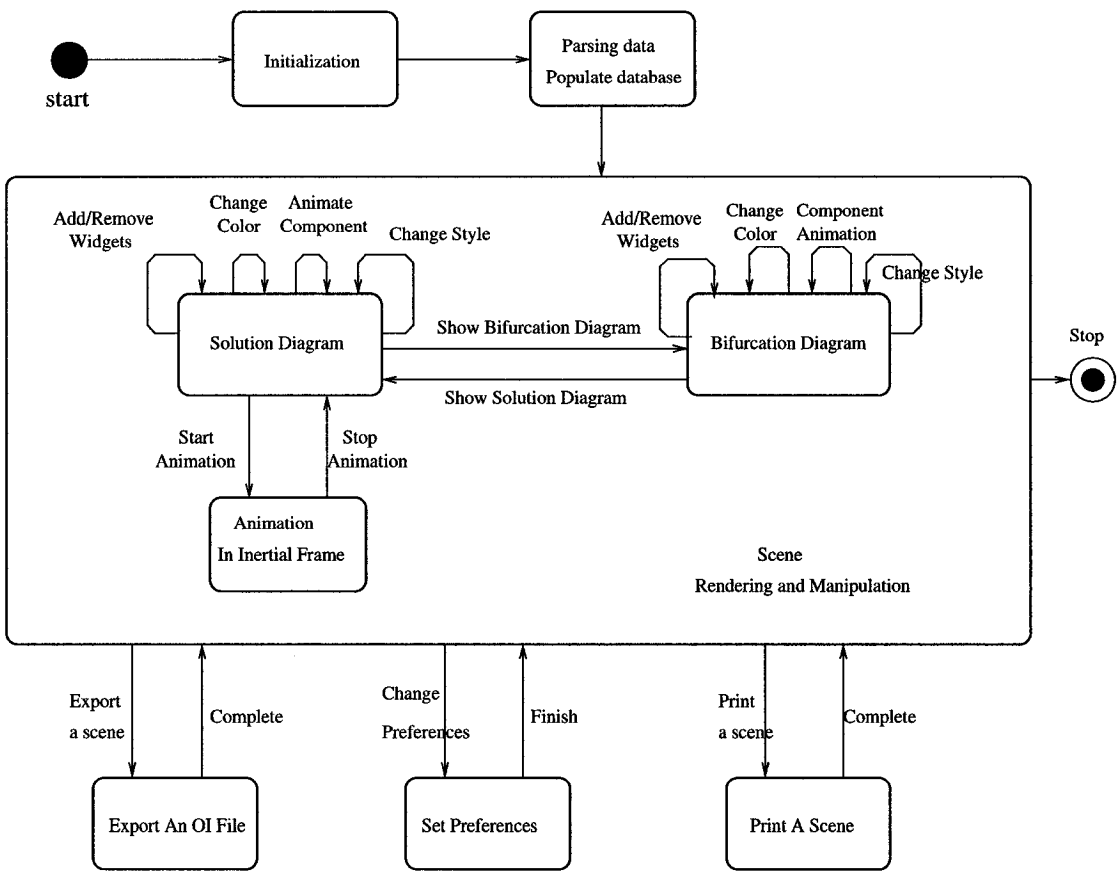


Figure 45: System state diagram

Figure 45 shows the various states which the system can progress through, on an abstract level. There are six states for the system, if we don't count the start and stop states.

When the program starts to run, after the system initialization, it parses the data from the AUTO data files and populates the database for the system. Then it enters the "Scene Rendering and Manipulation" state to present and manipulate the scene. The "Scene Rendering and Manipulation" state is the most important state of the system. It includes three sub-states, "Solution Diagram", "Bifurcation Diagram", and "Animation Inertial Frame". The system can stop running from this state.

The "Solution Diagram" sub-state is chosen as an example to give a detailed description of the "Scene Rendering and Manipulation" state.

In the "Solution Diagram" sub-state, the solution diagram is shown in the graphics render area. Any of the following events may trigger the state change:

- Add widgets to/remove widgets from the scene (reference plane, primaries, coordinate, background, legend);
- Change the style of the diagram;
- Change the coloring method;
- Animate a component;
- Show the bifurcation diagram;
- Start the animation in an inertial frame;
- Export a scene to a standard Open Inventor file;
- Print a scene;
- Change preferences.

The "Solution Diagram" has many sub-states. The first four groups of events lead it to one of its sub-states.

The "Show bifurcation diagram" event leads the system to the "Bifurcation Diagram" sub-state. In this sub-state, the bifurcation diagram is displayed. Similar to the "Solution Diagram" state, there are also many events which can lead to its state change.

When the "Start animation" event happens, the system is led to a new state, "Animation in an inertial frame". In this new state, the system animates the solutions or orbits in the inertial frame for the CR3BP. When this event is triggered, the system calls the data converter object to convert the data from the rotating frame to the inertial frame. Then it creates the scene and coordinates the motion of the three bodies.

When the "Export a scene " event happens, the system exports the current scene to an Open Inventor text format file, which can be viewed with Open Inventor SceneViewer. This function is useful for sharing the scene graph with those users who do not have PLAUT04 on their system.

When the "Change preferences" event happens, it allows a user to change the default settings for the system.

5.6.2 Graphic user interface design and implementation

Graphic user interface design

In order to satisfy the user requirements, the main GUI window is divided into two areas, the "command" area and the scene render area. The so-called "command" area is not a command line input zone, but a comprehensive name for the menu bar, dropdown list bar, the thumb wheels, and the pushbuttons, *etc.* In this graphical user interface, all commands are iconized so that a user can use the mouse to do the operations.

Figure 46 shows the layout of the GUI design. The center area is the render area for the scene graph. The areas around it are the "command" area, which is used to manipulate the scene graph and complete GUI operations.

The "command" area is composed of three subareas, the menu bar, the drop-down list bar, and the Pushbutton and Thumb Wheel Form.

The menu bar is composed of seven menus. Each of them represents a group of functions of the system. They control the states change of the scene graph. Switching between menu items can be done at any time as long as no other event sequence is running. The functions of each menu are described as follows,

- The *"Style" menu* allows a user to select the graphics style. The most favorite style is tubes. Other choices are curves, and surface.

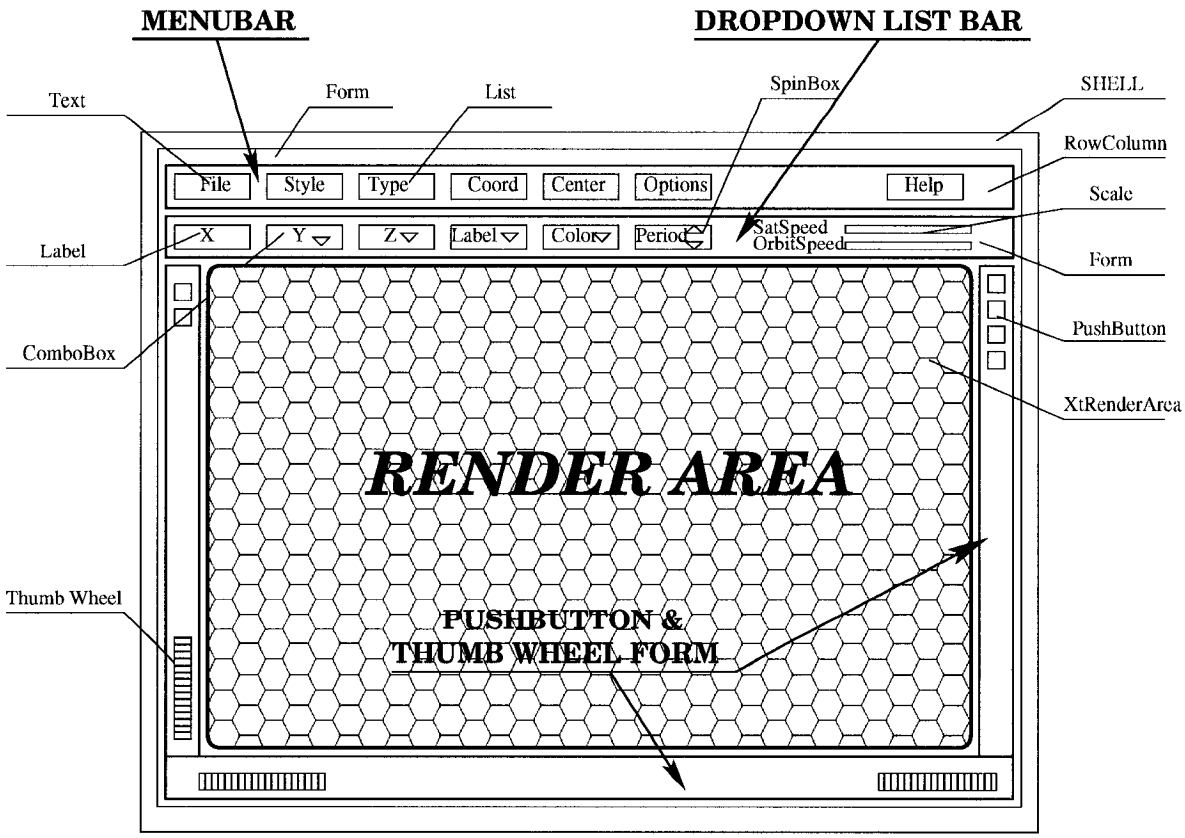


Figure 46: GUI component design

- The *"Type" menu* allows a user to choose between the solution diagram and the bifurcation diagram. In some cases, a user may choose the solution diagram as the default diagram, while in other cases, a bifurcation diagram tells more information. From this list, a user can switch between the solution diagram and the bifurcation diagram and extract more information from their comparison.

- The *"Coord" menu* provides functions for adding coordinate system axes and scales to the scene graph, or removing them from it.

- The *"Center" menu* is especially designed for the CR3BP. In this list, a user can animate the diagram in the inertial coordinate system. A useful feature is that a user can change the coordinate center of the system. There are three choices, "bary-centered", "large-primary-centered", and "small-primary-centered". In the general purpose version, this list is not shown.

- The *"Options" menu* focuses on adding widgets to the diagram as well as setting

preference variables. Optional widgets for a typical CR3BP solution diagram are "primaries", "reference plane", "orbit animation", "solution animation", *etc.* The default values of these choices can be set by choosing "preference" in this list.

- The last list is the *"Help" menu.* It provides detailed information about how to use PLAUT04.

The *drop-down list bar* is composed of six drop-down lists, one spin-box, and two scales. Each of them changes the scene graph's sub-state, namely, it can only change graphics' property, but does not make great changes to the structure of the graphics.

The functions of each menu are as follows,

- The *X, Y, Z lists* constitute the three axes for the 3-dimensional coordinate system. If a user wants to view a 2-dimensional graph, the Z list is disabled.

  The items for these three lists are the same. They are composed of the variables of a problem plus time for the solution diagram, and they are composed of the parameters of a problem for the bifurcation diagram. For example, in the CR3BP, the items for a solution diagram are $time$, x, y, z, x', y', and z'. However, the items for a bifurcation diagram may be $par[1], par[2], par[3], par[10], par[15]$, *etc.*, based on the user's problem specification.

- The *Label list* lets a user select the label to be drawn or animated.

- The *Color list* provides many different coloring methods for the scene graph.

- The *Period Spinbox* is used when a user animates an orbit in the inertial frame. In this case, a user can set the number of periods that needs to be animated before the system redraws the diagram on the screen.

- The *SatSpeed Scaler* and *OrbitSpeed Scaler* are used to adjust animation speed.

The *Pushbutton and Thumb Wheel Form* contains several pushbuttons and three thumb wheels for the zooming in/out and the rotating of the scene graph.

- The *2D/3D pushbutton* controls the switching of the graphics between 2D and 3D.

- The *Picking button* is used to pick a point in the diagram. This will lead to a pop-up window, where the raw data and related Floquet Multipliers of the point will be shown.
- The three Thumb Wheels are used for the zooming in/out and rotating of a scene graph.

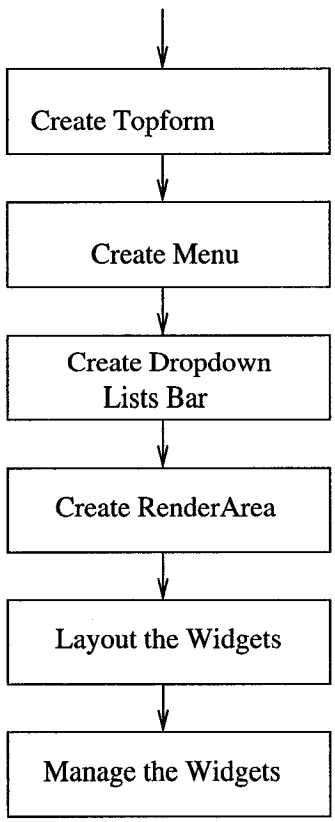


Figure 47: GUI flow chart

Graphic user interface implementation

As mentioned before, in the implementation of the GUI, many Motif Widgets are used. The main window of the design is implemented using a Motif Form class. The Form widget provides many controls over the placement and sizing of the widgets it manages. A Form can lay out its children in a grid-like manner or it can allow its children to link themselves to one another in a chain-like fashion. It is one of the most important Motif manager widgets.

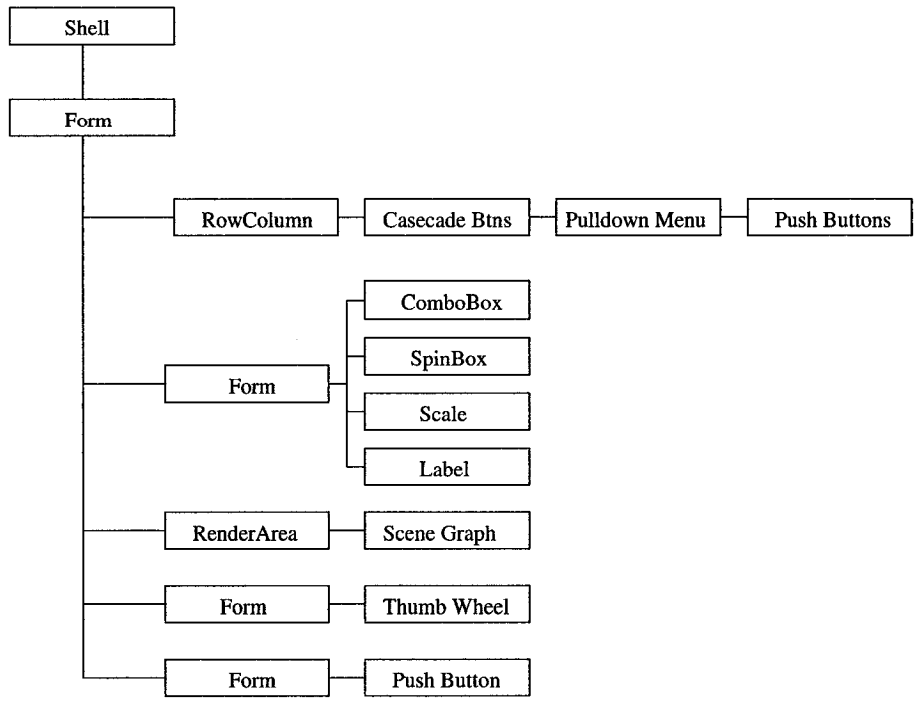


Figure 48: GUI parent-child relationship between widgets

The menu bar is implemented using a Motif RowColumn widget. This RowColumn widget has a convenient creation function, XmCreateMenuBar(), for users to create a menu bar easily. The menu bar is created and placed along the top of the application window, and seven CascadeButtons (file, type, style, coord, option, center, and help) are inserted as its children. Each of the CascadeButtons has a Pulldown menu panel associated with it. The menu bar, the seven CasecadeButtons and the associated Pulldown menus constitute the GUI menu system.

Another Form widget is used as the parent widget to lay out the drop-down lists, the SpinBox, and the Scales. The Motif ComboBox widget is an ideal selection to implement the drop-down list. The SpinBox widget is used to create the Period SpinBox. The Motif Scale widget is used as the Sat Speed and Orbit Speed adjustment scalers.

Open Inventor provides an Xt widget, SoXtRenderArea, which provides an easy way to integrate the Motif GUI and Open Inventor rendering area. The integration of Motif and Open Inventor is discussed further in Section 5.5.5.

Figure 47 shows the flow chart of the GUI implementation. The parent-child relationship between widgets is depicted in Figure 48.

5.6.3 Data manipulation component design and implementation

The data parser is concerned with the data parsing. The AUTO dataset is stored in its specific format. It includes great amount of information. However, not all of it is useful in the visualization. Before the graphical object can use it, the parser is used to parse useful information from AUTO datasets and the results are sent to the dataset object.

The dataset object stores and manages the data in a linked-list. The dataset object is also in charge of the converting of the data from the rotating system to the inertial system.

The basic unit for the solution file is a solution. The following C/C++ struct is used to represent it.

```
struct solution{
  long position;
  int nrowpr;
  int branch;
  int point;
  int type;
  int label;
  int new_label;
  struct solution *next;
};
```

The struct saves some basic information of the solution file. It does not really read the raw data into the memory. When necessary, PLAUT04 uses the position information to look for the data in the data file. This design avoids storing a lot of useless data in memory. In addition, it uses a random search method to read the file, which also increases the file reading speed.

Key data related to the graphical rendering, such as the number of points in the dataset, in a solution family, and in a particular solution, *etc.*, are saved in struct SolNode and BifNode, which are utility structures for the graphical object. The graphical object uses them to decide where and how to obtain the data. This design further increases the rendering efficiency of the graphics.

The data parser extracts the solution information and saves it in the solution struct. The pseudocodes for parsing the solution file are as follows.

```
Set head = new solutionNode
Read head->component
Set currentPtr = head

Read inputValue
WHILE NOT EOF{
    Set newNodePtr = new solutionNode
    Set newNodePtr->component = inputValue
    Set currentPtr->next = newNodePtr
    Set currentPtr = newNodePtr
    Read another inputValue
}
Set current->next = NULL
```

The codes of the data parser for the solution are almost the same as that used in creating a linked list.

The pseudocodes for parsing the bifurcation file are listed as follows.

```
    set numBranches = 0;
    set totalPoints = 0;
    set numPtInBranch = 0;
    set last = 0;

    Read inputLine;
    WHILE NOT EOF {
        if(branch != 0){
            if(last == 0){
                numBranches++
                numPtInBranch++
            }else if (last == branch){
                numPtInBranch++
            }else{
```

```
            set the number of points of the current branch to numPtInBranch
            numBranches ++
            numPtInBranch = 0
        }
    }else if(last !=0){
        set the number of points of the current branch to numPtInBranch
        numPtInBranch = 0
    }else{
        - A comment line, skip it.
    }
    last = branch
    Read inputLine
}
set myBifNode component value
```

Another important part of the data manipulation object is the data converter method. We have discussed the algorithm in Chapter 3.

5.6.4 Graphic object design and implementation

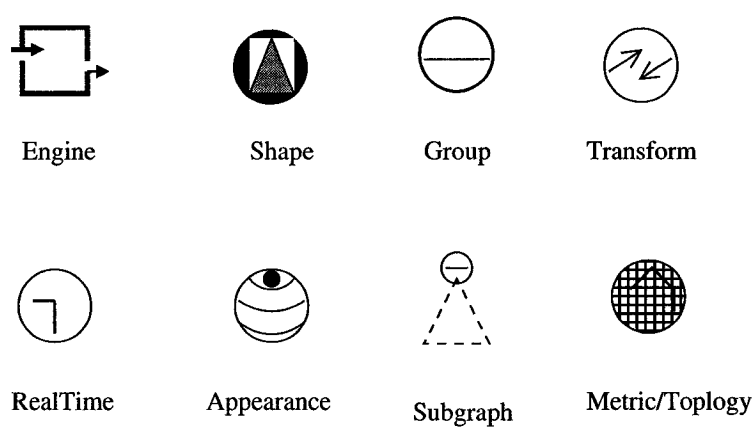


Figure 49: Scene graph symbols

The graphical object generates the graphics and maps it to the screen. Open Inventor toolkits are used in the graphical rendering. In this section, the Open Inventor scene graph node diagrams are used to describe our design. Figure 49 shows the symbols used in the scene graphs.

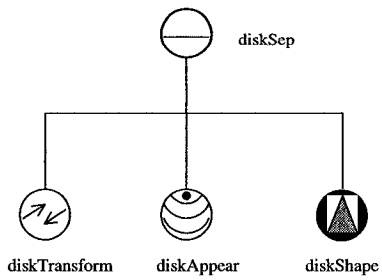


Figure 50: Reference plane node

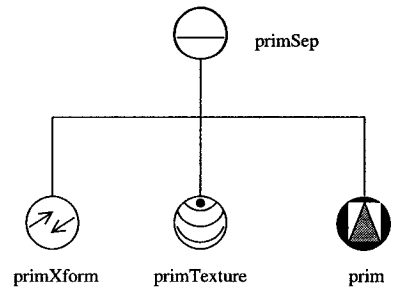


Figure 51: Primary node

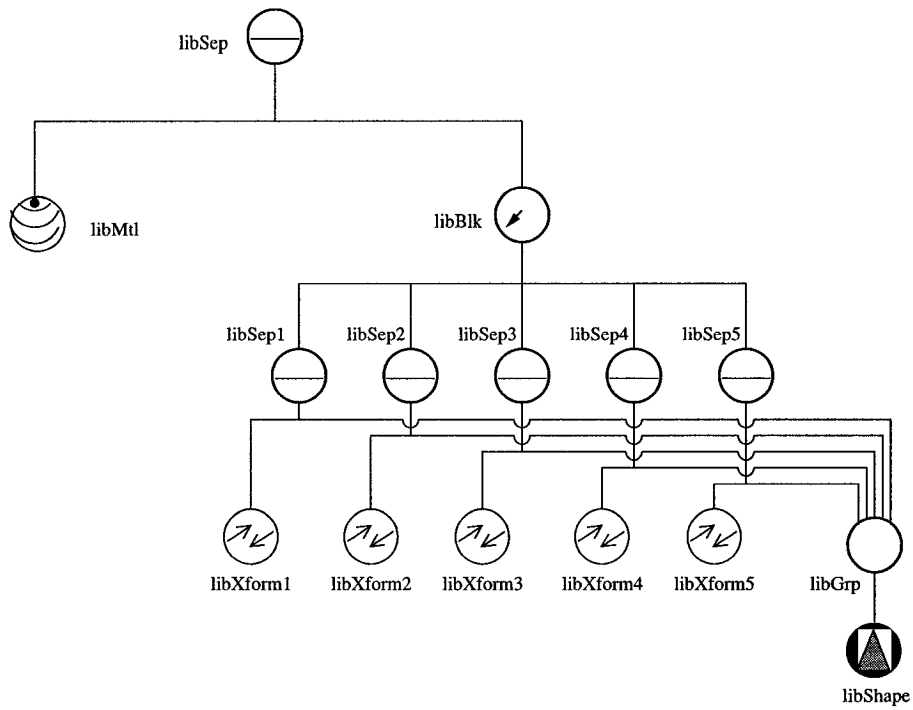


Figure 52: Libration point node

In the graphical objects, the widgets designed to satisfy the user's requirements are: solution/bifurcation scene widget, reference plane widget, coordinate system widget, libration points widget, and legend widget. Each of these widgets is composed of many Open Inventor nodes. For example, the reference plane widget is the simplest widget in our design, which has the structure shown in Figure 50. It includes an Open Inventor SoSeparator node, a SoTranform node, a SoMaterial node, and a SoShape node. The SoSeparator node is the root for the reference plane widget. It separates this node from the others, so that the properties defined by others do not have influence on the reference plane. The SoShape node defines the shape of

the reference plane and the way to draw it. The SoMaterial node sets the material properties such as texture, color, *etc.*, for the reference plane. The SoTransform node defines where the reference plane should be put and how to put it there.

Figures 51, 53, and 54 give the Open Inventor database structure for the other widgets. These widgets are much more complex than the reference plane widget. The solution and bifurcation scene widgets use other widgets and some advanced techniques (for example, engines, blinkers, sensors, *etc.*,) to animate solutions.

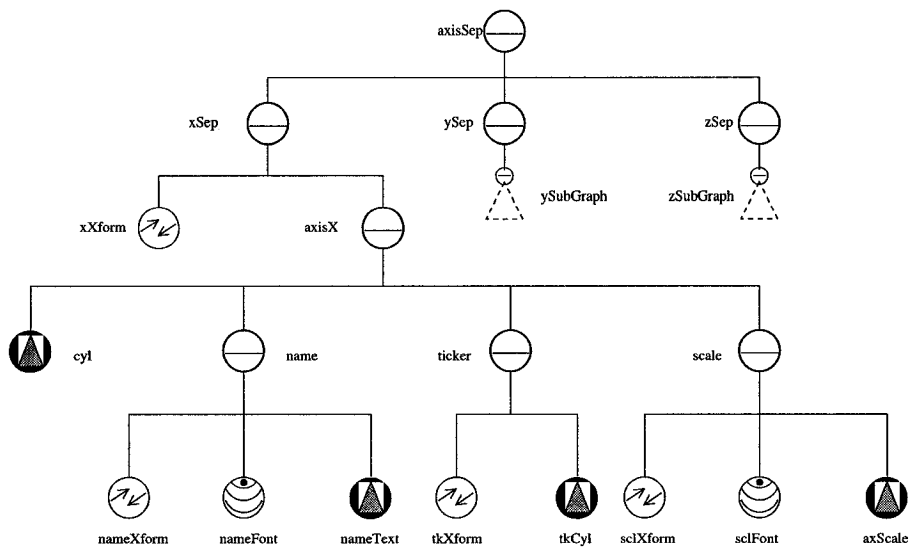


Figure 53: Coordinate system node

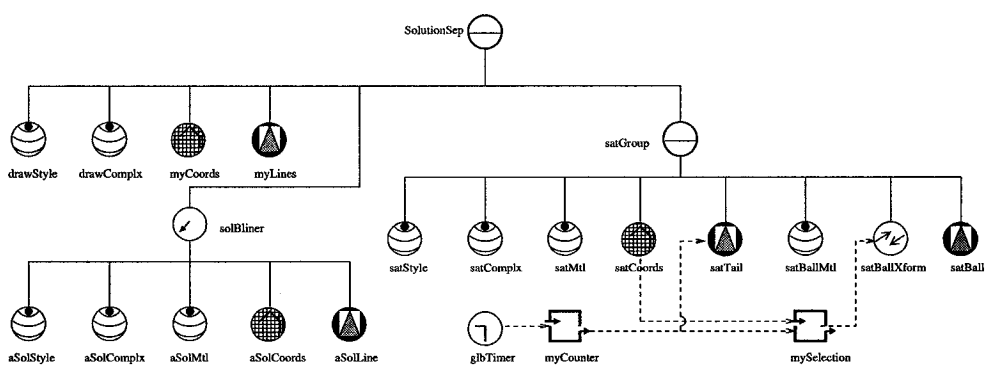


Figure 54: Solution node

In Figure 52, the five libration points have the same shape, color, and texture, and the only difference between them is that they are put in different locations. So the shape node and the material node are shared by the five libration points. In this

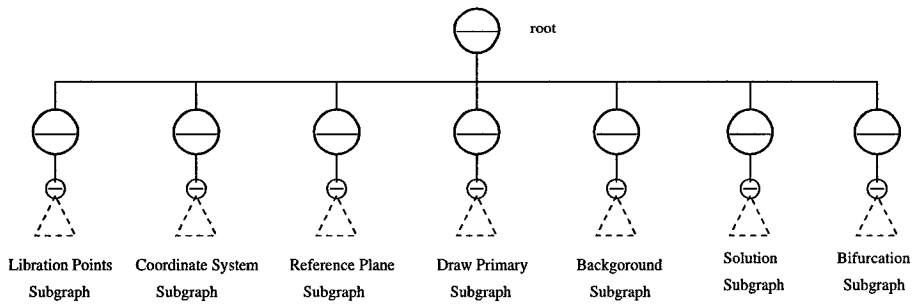


Figure 55: The full scene graph architecture

case, the shape node has five parents. Shared instancing offers database and program economy since objects can be reused without duplicating them.

Figure 53 shows the scene graph node for the coordinate system. It is created in three steps. In the first step, we create the three unit length axes, x-axis, y-axis, and z-axis, and different colors and textures are assigned to each axis. Then they are integrated into one unit coordinate system. In the last step, the unit coordinate is scaled and transformed so that it can fit the size of the scene graph and is put in the right position.

The most complicated and important scene graph is the solution scene. Figure 54 is a simplified version of what we used in PLAUT04. Nodes related to coloring scheme change, component animation, solution style change, *etc.*, are omitted. In this scene graph, two engines are used so that we can animate solutions in the inertial system. The first engine, "myCounter", receives input from the global timer, and starts counting from a minimum number to a maximum number. The step value indicates how the timer counts. The frequency input specifies the number of min-to-max cycles. The output is taken as the input of the selection engine, which controls the data to be drawn in the scene. This engine structure is the base of the animation. In the real code, we have to synchronize the motion of the primaries, the infinitesimal, the coordinate, and the reference plane, so the codes are more complicated than this design.

Having the sub-graphs prepared, it is easy to obtain the entire scene graph (see Figure 55).

The following presents the pseudocodes for creating the solution scene graph. Many details are omitted so that we can see the architecture more clearly.

```
Initialize scene root
```

```
If Opt_Normalize Data is TRUE
    - Normalize Data
If animateInInertialFrame is TRUE
    - Converter Data To Inertial Frame
    - animate the solution scene in inertial Frame
If Opt_DrawSolution is TRUE
    If Opt_DrawLibPoints is TRUE
        - create the libration point scene
        - add it to the root.
    If Opt_DrawCoordinates is TRUE
        - create the coordinates
        - add the coordinates to the root.
    If Opt_DrawReferPlane is TRUE
        - create reference plane
        - add the reference plane to the root
    If Opt_DrawPrimaries is TRUE
        - create primaries
        - add the primaries to the root
    If Opt_DrawBackgroud is TRUE
        - create background
        - add background to the root

    - create solution scene graph
    - add the solution scene to the root
else
    - create widgets for the bifurcation scene
    - add the widgets to the root
    - create the bifurcation scene graph
    - add the bifurcation scene graph to the root.
```

The pseudocodes show us the structure of the solution scene graph. The function prototypes for creating those widgets are listed below.

void normalizeSolData();

This function is used to normalize the solution data. Sometimes, a user may want

to rescale the original data to [0, 1]. It is exactly what this function does.

*SoSeparator * createDisk(float where[], float scaler);*

This function is used to create a reference plane widget. The position and the size of the disk should be decided in advance, and then a reference plane widget is returned.

*SoSeparator * createLibrationPoint(float mu, float dis, float size, char *texture-FileName);*

This is a function for creating the libration points. When this function is called, the first three arguments must be initialized, and the last one tells the procedure where to look for the texture for the libration points. "mu" is the mass ratio, "dis" is the distance between the two farthest points in the scene, and "size" is the size of the libration points. This function returns a libration point widget object.

*SoSeparator * createPrimary(double mass, double pos, float size, char *texture-FileName);*

This is a function for creating the primaries. It has four arguments. The "mass" is the mass of the primary. The "pos" tells the system where the primary should be drawn. The "size" decides how big the primary should be. The last argument is the file name for the texture used for this primary. This function returns a primary widget.

*SoSeparator * createStarryBackground(char * bgFileName);*

This is a function for drawing a starry background. This function takes a picture as a background for the scene graph. It reads a picture from a file, then converts it to an Open Inventor object and returns the object.

*SoSeparator * renderSolution( );*

This function creates the solution scene.

*SoSeparator * createSolutionInertialFrameScene(float dis);*

This is a function for creating a solution scene in the inertial frame. It calls the "convertDataToInertialSystem()" first and then uses the converted data to animate the motion of the three bodies in space.

*SoSeparator * createBifurcationScene();*

This function is used for creating a bifurcation scene.

5.6.5 Integrating Open Inventor and Motif

In most cases, a user wants to see the graphics and manipulate it. This requires that an application not only draws lively graphics, but also provides an easy-to-use method to manipulate the graphics and change variables related to the graphics. In other words, a useful graphic application needs not only a perfect way to view the graphics, but also has a user friendly interface to allow a user to change it.

As mentioned before, Open Inventor is used to create the 3D graphics and do the animation in our software. However, like most 3D applications, Open Inventor ends up using 3D graphics primarily in one or more "viewing" windows. In most cases, the graphical user interface aspects of an application use standard 2D interface objects such as pulldown menus, dropdown lists, slider bars, pushbuttons, and dialog boxes. Creating and managing these common user interface objects is where Motif, Tcl/Tk, and Qt perform well. Many reasons push us to take Motif as our choice in the GUI building. First of all, Motif is the X Window System's industry-standard interface for user interface construction. It is the most popular toolkits for building GUI for Unix and Unix like systems. Second, it is well supported by all desktop environments, such as KDE, Gnome, *etc.* Furthermore, the nice look-and-feel, well-documented, standard widget sets, make it easy to learn and convenient to use. Although Open Inventor provides simple mechanisms for handling events, such as key press or mouse movement, it would still be very complicated and time consuming to use them to build a powerful GUI. No one really uses them to program the GUI in real application.

The Open Inventor SoXtRenderArea Component provides us an Xt render area for displaying a scene graph. This library is an Inventor C/C++ wrapper around a Motif-compliant widget. We can layer components in a window with other Motif widgets using standard layout schemes, such as bulletin boards, and form widgets. This Inventor Xt Component provides an easy way to integrate Open Inventor and Motif.

In the render Area, the SoXtRenderArea is an Xt widget that performs OpenGL rendering. When it receives X events, it translates them into SoEvents, which are then passed to the scene manager for handling.

We have shown the key manager and primitive widgets that make up the GUI of our application in Figure 46. The following are the source codes for integrating the GUI and the Open Inventor render area.

```
    // build the toplevel widget
    topform = XtCreateWidget("topform", xmFormWidgetClass, parent,NULL, 0);

    // build menubar
    Widget menubar = buildPulldownMenu(topform);

    // build list carrier for the x, y, z, and label lists.
    Widget listCarrier= XtCreateWidget("ListCarrier",
                xmFormWidgetClass, topform, NULL, 0);

                ......

   // create RENDER AREA FOR THE graphics.
    renderArea = new SoXtExaminerViewer(topform);
    renderArea->setSceneGraph(sceneGraph);

                ......

    // LAYOUT THE FORM. LAYOUT ALL THE WIDGETS.
    // Positioning the menu bar.
    n = 0;
    XtSetArg(args[n], XmNtopAttachment,     XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNleftAttachment,    XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNrightAttachment,   XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNbottomAttachment,  XmATTACH_NONE); n++;
    XtSetValues(menubar, args, n);

    // Positioning the listCarrier.
    n = 0;
#ifdef LIST_UNDER_MENUBAR
    XtSetArg(args[n], XmNtopAttachment,     XmATTACH_WIDGET); n++;
    XtSetArg(args[n], XmNtopWidget,         menubar        ); n++;
    XtSetArg(args[n], XmNbottomAttachment,  XmATTACH_NONE); n++;
```

```
#else
    XtSetArg(args[n], XmNtopAttachment,      XmATTACH_NONE); n++;
    XtSetArg(args[n], XmNbottomAttachment,   XmATTACH_FORM); n++;
#endif
    XtSetArg(args[n], XmNleftAttachment,     XmATTACH_FORM); n++;
    XtSetArg(args[n], XmNrightAttachment,    XmATTACH_FORM); n++;
    XtSetValues(listCarrier, args, n);

    // Positioning the Render Area
    n = 0;
#ifdef LIST_UNDER_MENUBAR
    XtSetArg(args[n], XmNtopAttachment,      XmATTACH_WIDGET); n++;
    XtSetArg(args[n], XmNtopWidget,          xAxisList       ); n++;
    XtSetArg(args[n], XmNbottomAttachment,   XmATTACH_FORM   ); n++;
#else
    XtSetArg(args[n], XmNtopAttachment,      XmATTACH_WIDGET); n++;
    XtSetArg(args[n], XmNtopWidget,          menubar         ); n++;
    XtSetArg(args[n], XmNbottomAttachment,   XmATTACH_WIDGET); n++;
    XtSetArg(args[n], XmNbottomWidget,       xAxisList       ); n++;
#endif
    XtSetArg(args[n], XmNleftAttachment,     XmATTACH_FORM   ); n++;
    XtSetArg(args[n], XmNrightAttachment,    XmATTACH_FORM   ); n++;
    XtSetValues(renderArea->getWidget(), args, n);

    // manage the children
    XtManageChild(menubar);
    XtManageChild(listCarrier);
    XtManageChild(topform);
```

The topform, the Motif FORM widget above the SHELL, is the main widget that contains the interface layout. The menuBar, listCarrier, and the renderArea are positioned above the topform, and all the other widgets are rooted at them. This organization makes the GUI clean and well-organized.

5.7 Running and testing

A high quality, stable package is always based on a thorough test scheme and extensive testing. In this part, a variety of test cases are outlined for PLAUT04. These test cases are geared to examine the functionality of our software components as extensively as possible, yet within a reasonable amount of time. In order to maximize the stability of our software, we have provided test cases which follow the normal operation of our program, along with some to handle exceptions.

5.7.1 Starting the program

Before we talk about the test cases, we'd rather discuss how to start the program. We suppose that both the hardware and software environments are satisfied, PLAUT04 has been installed in the system, and the environment variables have been set. More detailed description about the installation is described in Appendix B. The related software and hardware environments for running PLAUT04 are described in Appendix B too.

There are three methods to start the program. The simplest method is by typing "plaut04" in the command line of a Linux console window as follows,

```
$ plaut04
```

After keying in this command, the program will start to run. It will first look for the resource file in the current directory to get the specified resource settings. If there is no resource file in the current directory, the program will access the default directory and use the default resource file. If both are not found, it will use the internal settings. After that the system will look for the default AUTO data files, namely, fort.7, fort.8, and fort.9, in the current directory. If none of them exists, an error message will be given. If the above two steps execute successfully, we should see the GUI and the scene graph. Figure 56 is a snapshot of the screen. From the above discussion, we can see that this first method to run PLAUT04 is used to show the default AUTO data files under the default settings.

The second method to start PLAUT04 is used when we want to view a specific AUTO data file in the current directory. In this case, we need to type the command with an argument, the data file name. For example, if we want to see AUTO data

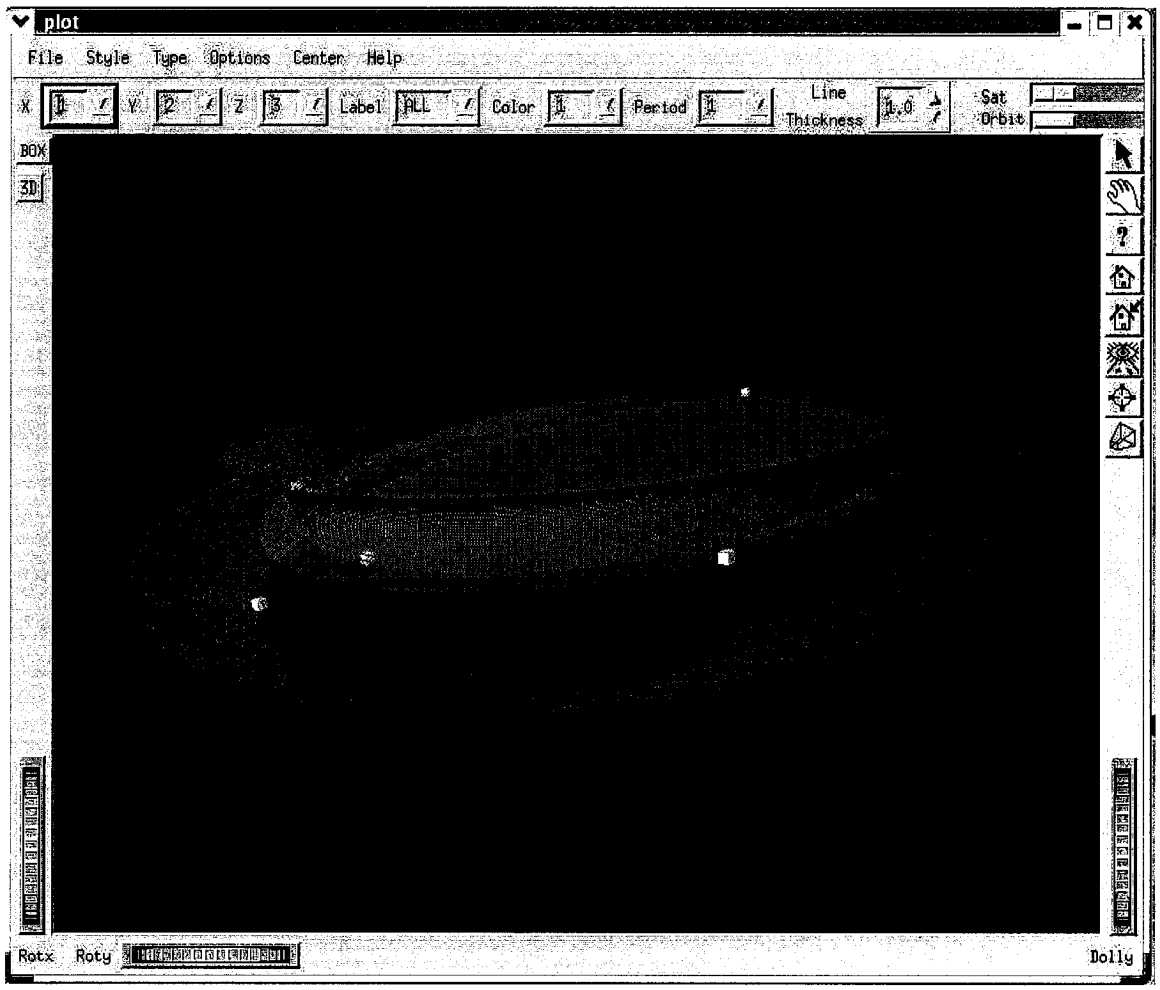


Figure 56: A snapshot of the GUI

files "s.H1", "b.H1", and "d.H1" which are stored in the current directory, we need type the command as follows.

```
$ plaut04  H1
```

We can also run PLAUT04 to view a specific AUTO data file in other directories. In this case, both the path to the directory and the file name are required. For example, if we have "s.P1", "b.P1" in the directory "/home/he/myR3B/me", the following command should be used to view it.

```
$ plaut04 /home/he/myR3B/me     P1
```

Note that the two arguments are separated by space.

For more detailed operation about the GUI, please refer to Appendix B. Actually, almost all the operations can be found in the GUI menu bar by name. An on-line help also provides guidance for using the functions of the menu bar and the drop-down lists.

5.7.2 Test case specifications

The test cases are grouped into two parts. In the first part, the testing for the starting, shut-down, and initialization of the system is discussed. In the second part, the testing for the major functions of the system is discussed. We group the test cases based on the functionality of the system. There are so many different combinations that we could not enumerate all of them, so only some of the most important ones are listed here.

Starting and initialization test cases

This section deals with testing the starting of PLAUT04. It helps to make sure that the program can be started properly when the arguments are provided correctly. If the arguments are not provided in the right way, an error message should be printed.

This section also deals with the system initialization. It is focused on the resource file testing. The purpose of this test case is to ensure that the resource file can be correctly read. In the case that the resource file is not well written, the system should still work and the error values should be automatically discarded.

Test description

Starting of the program

Initial system state

No state (Nothing has been created, used or started).

Input values

In this test, there are three types of input. In the first case, there is no input argument. In the second case, one argument is provided. In the third case, two arguments , namely, the path and the filename, are provided.

Execution sequence

1. Resource file is created.

2. AUTO data file is created.

3. Starting command is issued.

Expected results

1. The package should be able to read the resource file and AUTO data files correctly.

Exceptions

1. If more than four arguments are typed, an error message should be issued and the system should not start.

2. If no objects is found, error message should be printed out.

GUI functionality test cases

The test cases in this section concern with functionalities that can be performed from the menu bar of the GUI.

I. "Type" Menu test case

Test description

Check to see if the available selections in the list work well. There are two choices in this list. One is "solution", the other is "bifurcation". Based on the default value, the system will choose which one to be shown.

Initial system state

The system is started normally and both "s.xxx" and "b.xxx" are read correctly.

Input values

1. type = SOLUTION, *i.e.*, the solution file is selected to be drawn.

2. type = BIFURCATION, *i.e.*, the bifurcation file is selected to be drawn.

Execution sequence

1. The global variable "graphTypeWidget" is set to "SOLUTION".

2. The method updateScene() is called to reload the data and update the graph.

3. The global variable "graphTypeWidget" is set to "BIFURCATION".

4. The method updateScene() is called to reload the data and update the graph.

Expected results

1. The old value of the global variable "graphTypeWidget" is substituted correctly

2. The scene graph is updated correctly, *i.e.*, when SOLUTION is selected, the correspondent solution diagram should be drawn in the graph area, otherwise, the bifurcation diagram should be drawn in the graph area.

3. All fields contain what they are supposed to have.

II. "Style" Menu test case

Test description

This test focuses on the "Style" list of the menu bar. It verifies the correctness of the available selected items in the list. There are three choices in this list, *i.e.*, "line", "tube", and "surface". By default, "line" is selected. A user can set up his own default value in the resource file.

Initial System State

The system is started normally and both "s.xxx" and "b.xxx" are read correctly.

Input values

In this test case, the input value is the variable "graphStyleWidget". It can be set to "LINE", "TUBE", and "SUFRACE". We can start from any one of the three values.

Execution sequence

We just give one sequence as an example for our testing. The other sequences are similar to this.

1. The global variable "graphStyleWidget" is set to "LINE".

2. The method updateScene() is called to reload the data and update the graph.

3. The global variable "graphTypeWidget" is set to "TUBE".

4. The method updateScene() is called to reload the data and update the graph.

5. The global variable "graphTypeWidget" is set to "SURFACE".

6. The method updateScene() is called to reload the data and update the graph.

Expected results

1. The old value of the global variable "graphStyleWidget" is substituted correctly

2. The scene graph is updated correctly.

The Execute sequence should be arbitrarily changed so that each branch is tested and to make sure the result is not sequence dependent.

III. "Options" Menu test case

Test description

In this test case, the "Options" list is tested. There are seven choices in this list, "solution change animation", "orbit change animation", "draw legend", "draw background", "draw reference plane", "draw primaries", and "draw libration point". The first two options control the animation of the solution diagram. So in the Bifurcation diagram, they should be disabled. Each of the last five options adds certain widget to or removes it from the diagram.

Initial system state

The system is started up normally and both "s.xxx" and "b.xxx" are read correctly.

Execution sequence

The following steps must be tested randomly. The result must not be influenced by the specific sequence which a user takes.

1. The global variable "graphWidget" is set to "draw reference plane".
   The method updateScene() is called to reload the data and update the graph.

2. The global variable "graphWidget" is set to "draw primaries".
   The method updateScene() is called to reload the data and update the graph.

3. The global variable "graphWidget" is set to "draw libration point".
   The method updateScene() is called to reload the data and update the graph.

4. The global variable "graphWidget" is set to "draw legend".
   The method updateScene() is called to reload the data and update the graph.

5. The global variable "graphWidget" is set to "draw background".
   The method updateScene() is called to reload the data and update the graph.

6. The global variable "graphWidget" is set to "Solution change animation".
   The method updateScene() is called to reload the data and update the graph.

7. The global variable "graphWidget" is set to "orbit change animation".
   The method updateScene() is called to reload the data and update the graph.

Expected results

The scene graph is updated correctly. In the above first 5 steps, after each step the correspond widget should be correctly added to or removed from the scene graph no matter what sequence is executed.

When each of the last two steps is chosen, the system should either start or stop the solution animation or the orbit animation.

IV. "Center" Menu test case

Test description

Check to see if the available selection items in the list work correctly. There are four choices in this list, *i.e.*, "rotating system", "bari-centered system", "large-primary-centered", and "small-primary-centered".

Initial system state

The system is started normally and both "s.xxx" and "b.xxx" are read correctly. In system default, the graph is shown in the rotating system.

Execution sequence

1. The global variable "graphCoordinateSystem" is set to "bari-centered".
   The method updateScene() is called to convert the data and redraw the graph.

2. The global variable "graphCoordinateSystem" is set to "large-primary-centered".
   The method updateScene() is called to convert the data and redraw the graph.

3. The global variable "graphCoordinateSystem" is set to "small-primary-centered".
   The method updateScene() is called to convert the data and redraw the graph.

4. The global variable "graphCoordinateSystem" is set to "rotating system".
   The method updateScene() is called to convert the data and redraw the graph.

Expected results

The scene graph is updated correctly, The data is automatically converted between the rotating system and the inertial coordinate system.

V. "File" Menu test case

Test description

Check to see if the available selection items in the list work correctly. The system should be able to export a graph into an Open Inventor text file. This text file can be read with Open Inventor SceneViewer.

Initial system state

The system is started normally and both "s.xxx" and "b.xxx" are read correctly. The data is drawn correctly.

Execution sequence

1. export2OI() is called.

2. write2OI() is executed and an Open Inventor text file is generated.

Expected results

In the current directory, a new Open Inventor text file is generated and the same graph should be drawn when it is opened with Open inventor SceneViewer.

5.8 Results

5.8.1 Creating solution diagrams

A solution diagram can be drawn using curves, tubes, or as a surface. Generally we suggest a user use curves to show a diagram. Use of curves is the simplest and the fastest way to show the solutions. Drawing the diagram by using curves takes less system memory and the graphics updating is so fast that the time can be ignored. This is especially the case for large datasets and complicated graphics. Figure 57 is an example of the solution diagram using curves. Figure 58 gives the corresponding tubes version. In general, it is much nicer to use tubes to show the diagram. However, it takes more time and system memory. Although creating a graph using tubes is not as fast as using curves, in most cases a user may not notice the difference between them. In some special situation, such as a graph that has many solutions and the dataset is huge, it will take some time for the system to deal with data parsing and calculation. A surface diagram is displayed for the dataset in Figure 59.

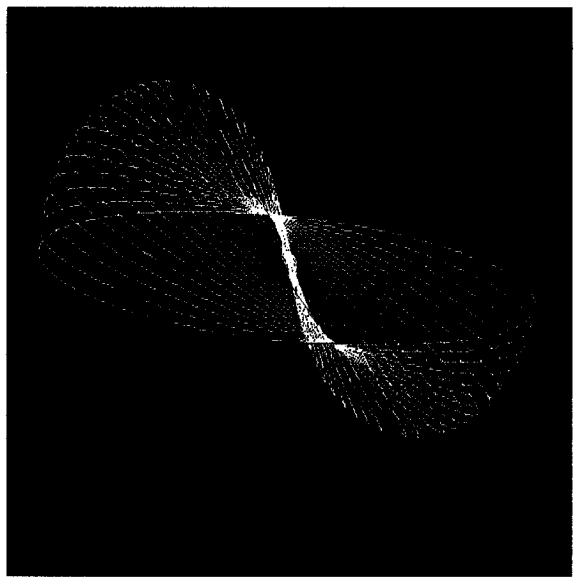


Figure 57: Drawing solutions using curves

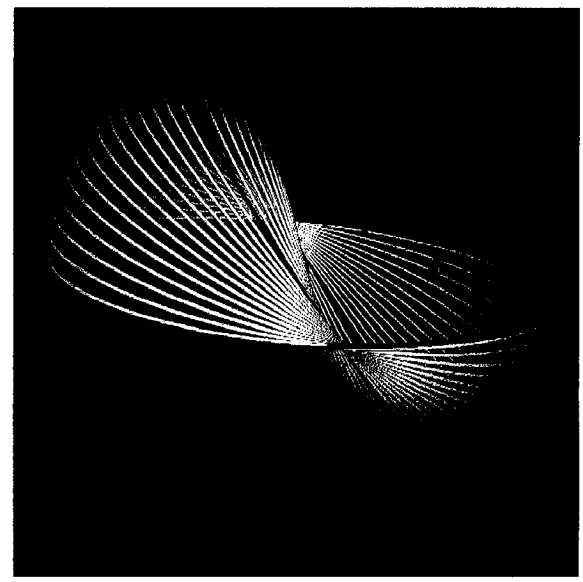


Figure 58: Drawing solutions using tubes

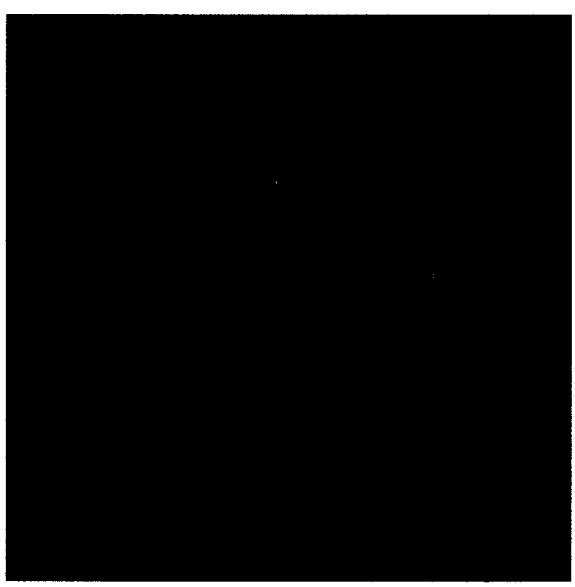


Figure 59: Drawing solutions as a surface

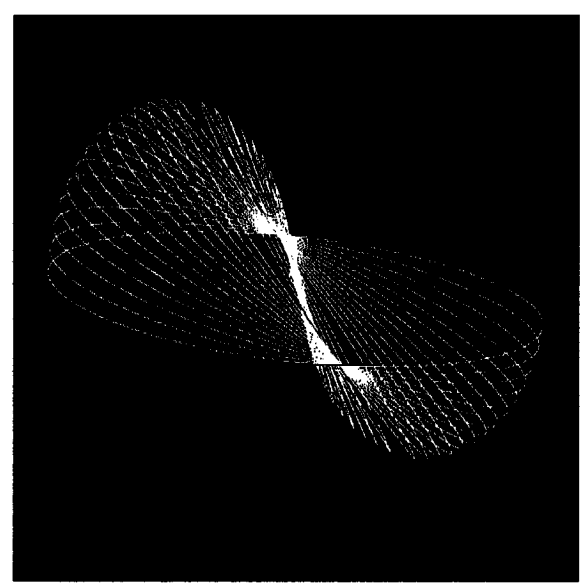


Figure 60: Drawing using Nurbs curves

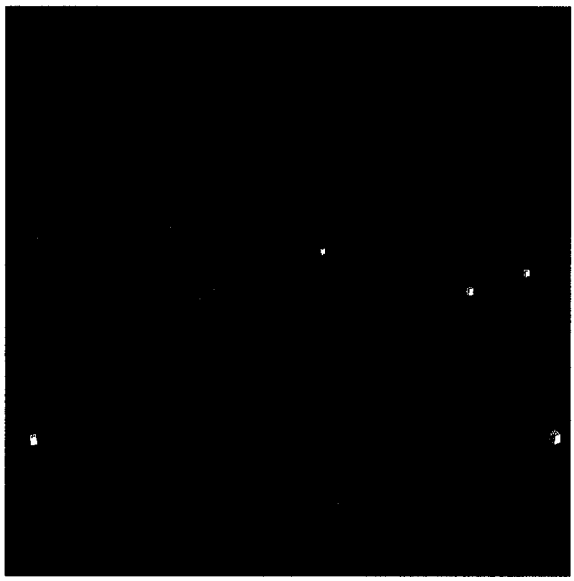


Figure 61: Drawing a bifurcation diagram using curves

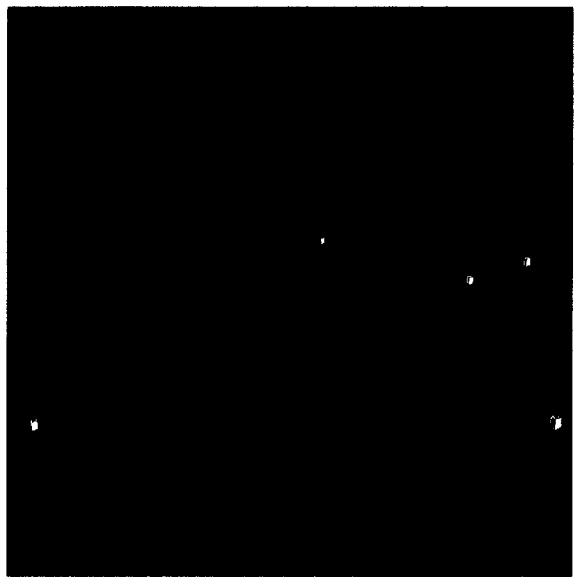


Figure 62: Drawing a bifurcation diagram using tubes

5.8.2 Creating bifurcation diagrams

AUTO bifurcation diagrams are rarely in the form of surface. So we don't implement a method to show a bifurcation diagram as a surface. Figure 61 and Figure 62 give bifurcation diagrams for the L3 family of the Earth-Moon system.

5.8.3 Animation of the motion of a satellite

Figure 63 and Figure 64 show the animation of the motion of a satellite in the rotating frame and in the inertial frame. The red points in the diagrams are satellites and the curves following them are their orbits.

5.8.4 Animation of a solution family

It is hard to draw an animation on paper. We only grab some of the interim screen shot to show the concepts. They are three approaches to animate solutions. The first approach is showing all orbits and animating the change of the orbits as depicted in Figure 65(a). The second is showing an interested orbit and animating the change of the others as shown in Figure 65(b). The third approach is showing the animation with no orbit specified as displayed in Figure 65(c).

Figure 63: Animation of the motion of a satellite in the rotating frame

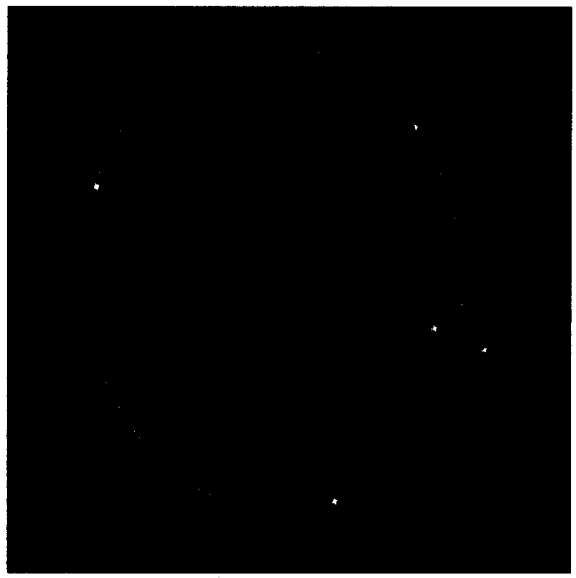
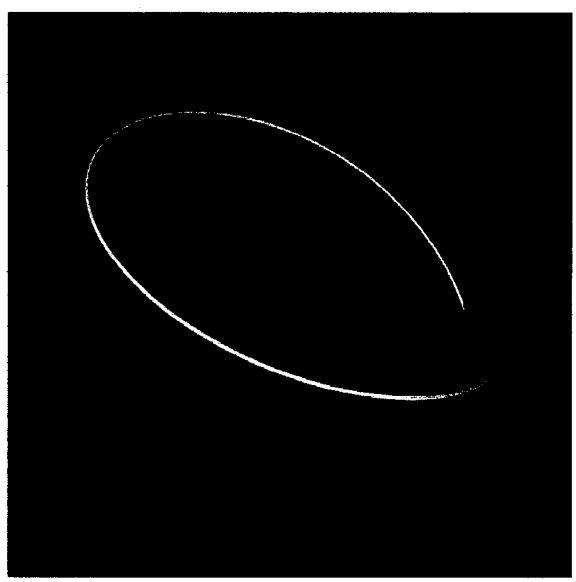


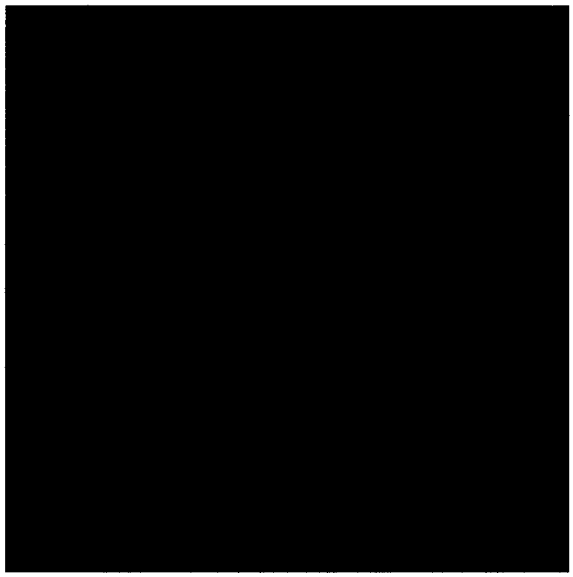
Figure 64: Animation of the motion of a satellite in the inertial frame

(a) Show all orbits



(b) Show a specified orbit

Figure 65: Solution animation

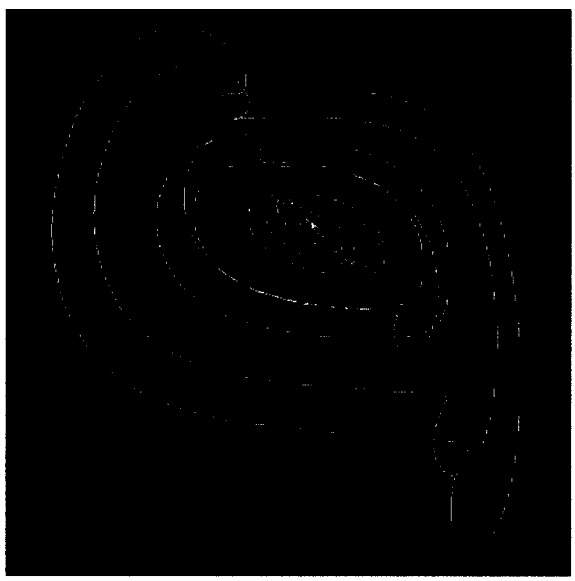


(c) No orbit specified

Figure 65: Solution animation (cont'd)

(a) Coloring by type

Figure 66: Coloring the diagram



(b) Coloring by family

(c) Coloring by label

Figure 66: Coloring the diagram (cont'd)

5.8.5 Coloring solutions

There are many ways to color a solution or bifurcation diagram. The basic ones are coloring by the type of the points, by the family of the solution/bifurcation, by point number, or by label. One can also color the solution/bifurcation diagram based on any column of the AUTO solution/bifurcation data. Furthermore, PLAUT04 allows a user to color the solution by using any problem-related parameters.

Figures 66(a), 66(b), 66(c), show the results of different coloring methods.

5.8.6 Picking a point in the diagram

The picking function allows a user to pick any point on the diagram. When a point is picked, a new window pops up. In this new window, the related data and the Floquet Multipliers of that point are shown. Figure 67 is an example of the picking function.

The above testing results show some typical functions of PLAUT04. For more examples, one can refer to the user manual.

5.8.7 Examples

PLAUT04 can be used to view any AUTO datasets. We give some examples of available datasets in this section. In these examples, the datasets are tested on a PC with the following specifications:

1. Hardware
   Processor: Intel Pentium 4
   CPU Speed: 1.6 GHz
   RAM: 384 Megabytes

2. Software
   Platform: Redhat Linux 9.0

Examples of large dataset

Example 1 and Example 2 show the capability of showing a large dataset using PLAUT04.

Example 1

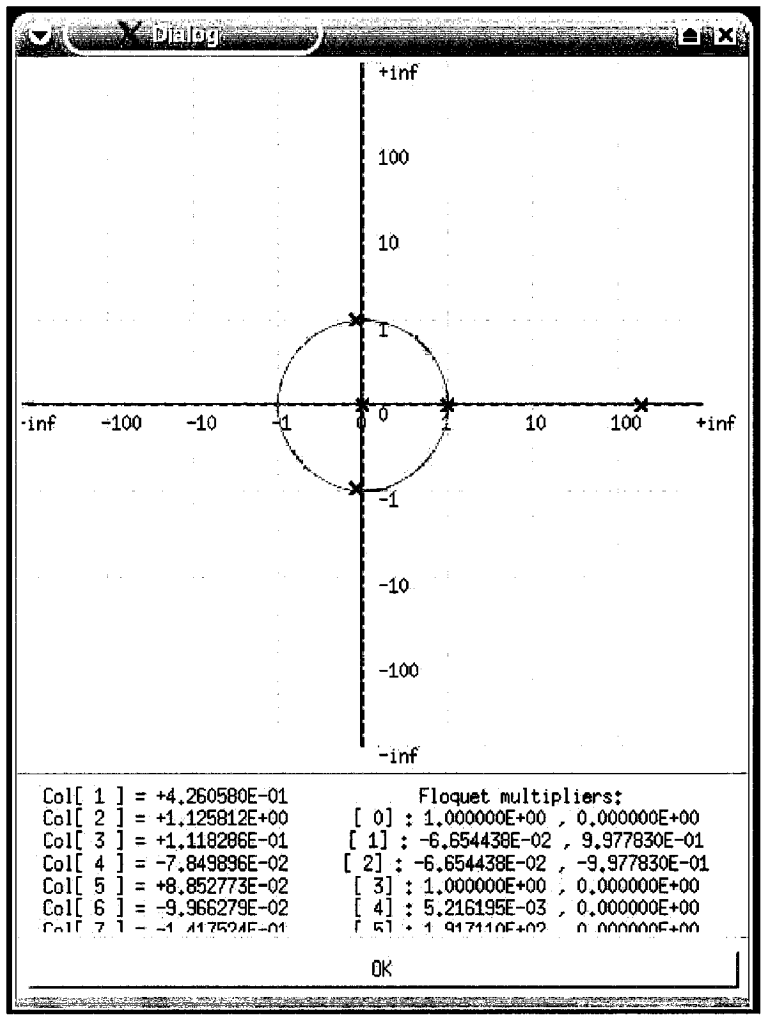


Figure 67: Picking a point

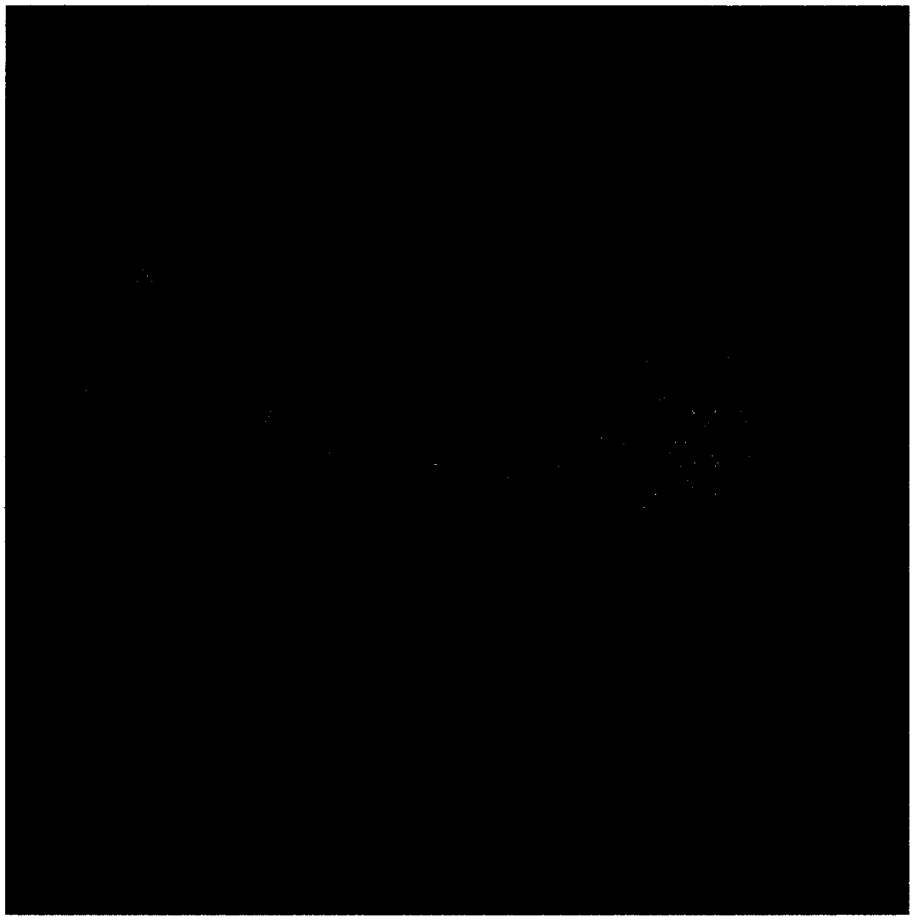


Figure 68: The solution diagram of the Lorenz problem

The data for this example is from the Lorenz problem. The size of the solution file is 27.83 Megabytes, the bifurcation file is 3.03 Megabytes, and the diagnostic file is 0.12 Megabytes. The total size of the dataset is 30.96 Megabytes.

From the time when the command is issued to the time when the picture appears on the screen, it takes approximately 12 seconds. Figure 68 shows the solution picture of this demo.

Example 2

The data of this example is from the CR3BP. It includes all periodic families emanating from the libration points of the Earth-Moon system. There are 18 branches, and 1437 solution orbits in the dataset. The size of the solution file is 137.41 Megabytes, the bifurcation file is 11.26 Megabytes, and the diagnostic file is 1.31 Megabytes. The total size of the dataset is 150 Megabytes.

From the time when the command is issued to the time when the picture appears

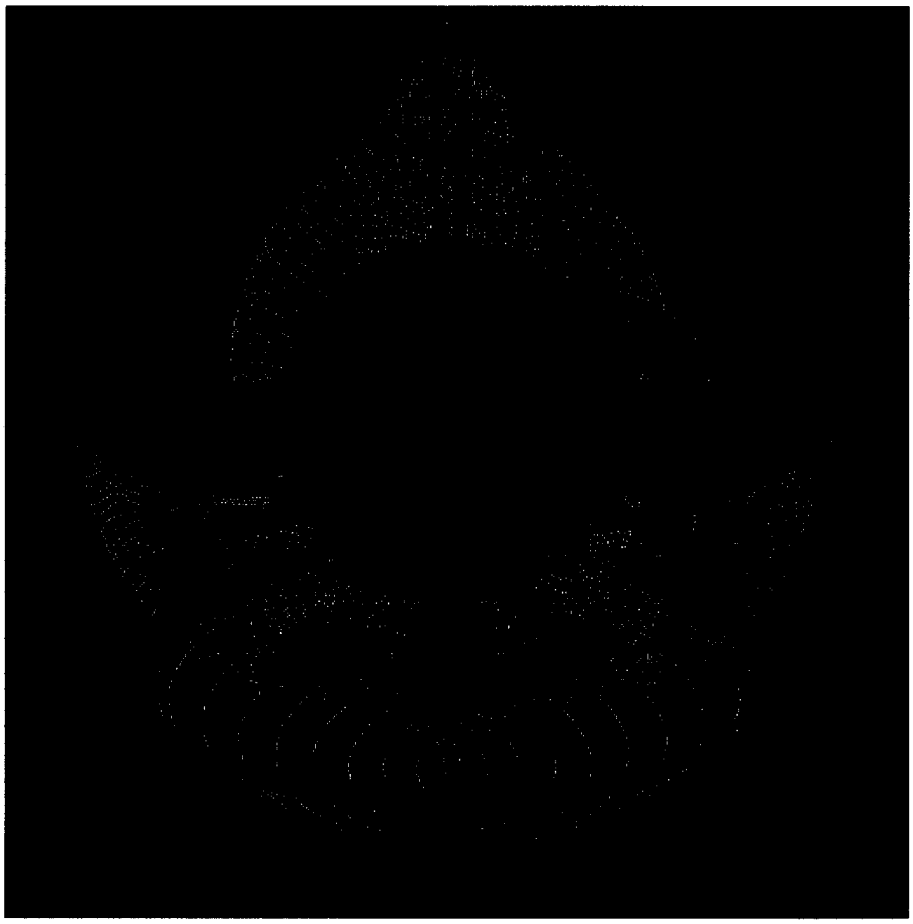


Figure 69: The solution diagram of the Earth-Moon system

on the screen, it can take up to two and half minutes. However, in most cases, it takes less than one minute. It depends on how many system resources are available. Most of the time is spent on parsing and loading data. This example is for testing purpose only. Because there are too many orbits, which overlap heavily, we cannot really get much useful information from the diagram. Figure 69 shows the result.

From these two examples, we can see that the performance of PLAUT04 is quite acceptable. It also shows that PLAUT04 is a robust tool. Example 2 includes 18 branches and 1437 orbits. This is much larger than that in most AUTO datasets.

Examples for problems with periodic solutions

In this section, some examples of periodic solution problems are given. All the data are from AUTO demo files.

Example 3

Figure 70: The solution diagram of AUTO demo `abc`

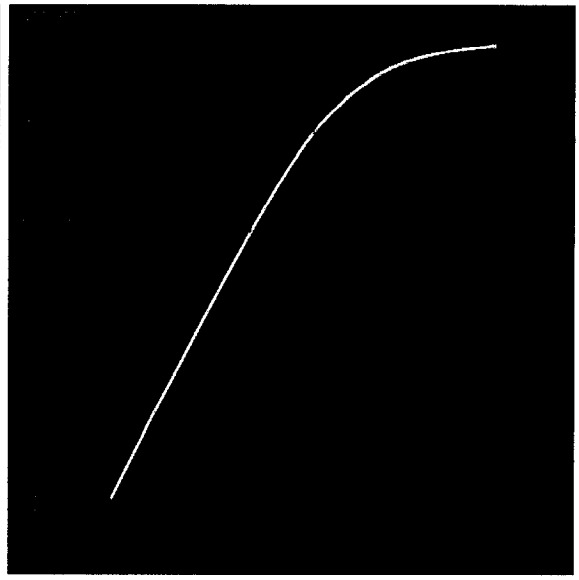


Figure 71: The bifurcation diagram of AUTO demo `abc`

Figure 70 and Figure 71 show the solution and bifurcation of AUTO demo `abc` (see AUTO Manual [24]).

Example 4

Figure 72 and Figure 73 show the solution and bifurcation of AUTO demo `pp2`.

Examples of BVP

Example 5 and Example 6 represent AUTO BVP demos.

Example 5

Figure 74 and Figure 75 show the solution and bifurcation AUTO demo `exp`.

Example 6

Figure 76 and Figure 77 show the solution and bifurcation of AUTO demo `bvp`.

Examples of parabolic PDEs

Example 7

Figure 78 and Figure 79 show the bifurcation diagrams of AUTO demo `brf` and `bru`.

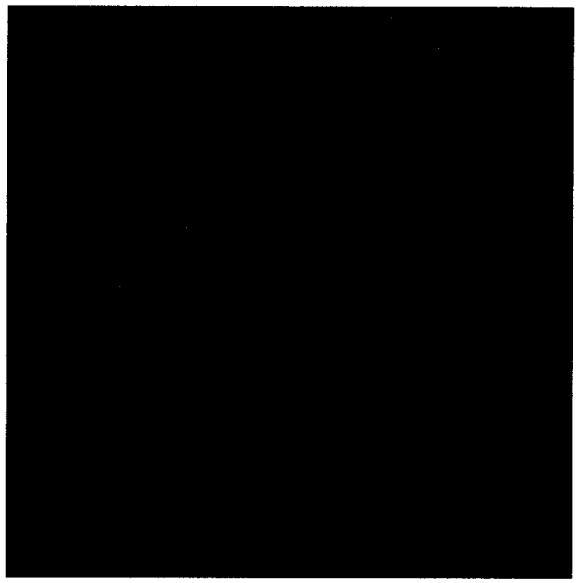


Figure 72: The solution diagram of AUTO demo `pp2`

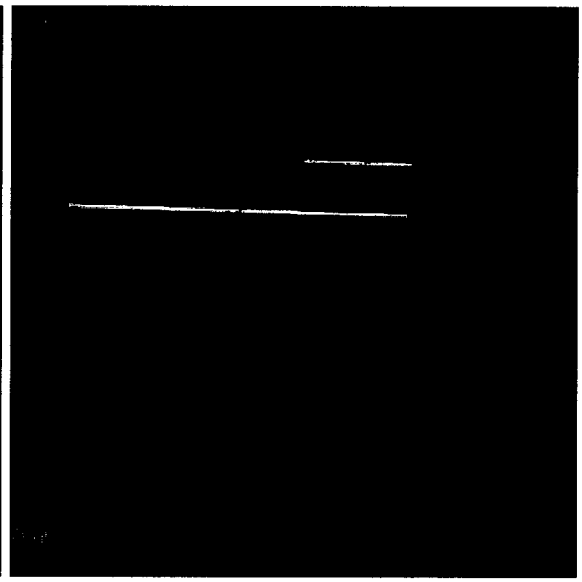


Figure 73: The bifurcation diagram of AUTO demo `pp2`

Examples of HomCont

Example 8

Figure 80 and Figure 81 show the solution diagrams of AUTO demo `kpr` and `cir`.

PLAUT04 can be used to view the solutions and bifurcation diagrams of virtually all AUTO applications.

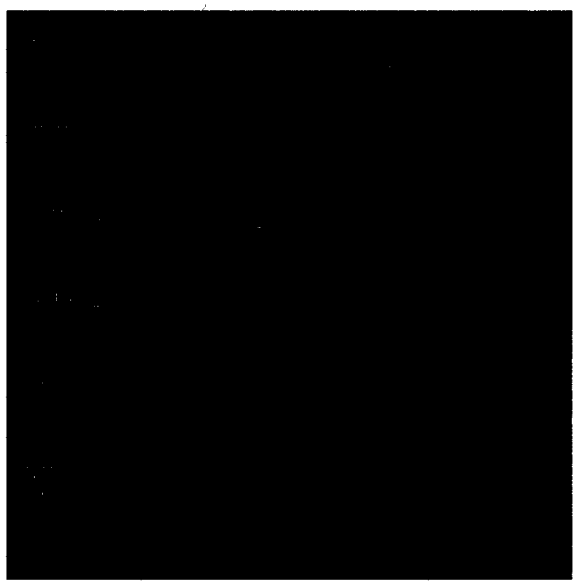


Figure 74: The solution diagram of AUTO demo `exp`

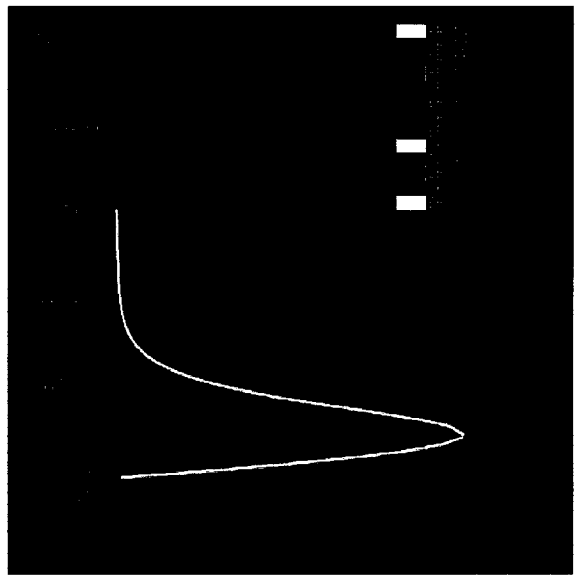


Figure 75: The bifurcation diagram of AUTO demo `exp`

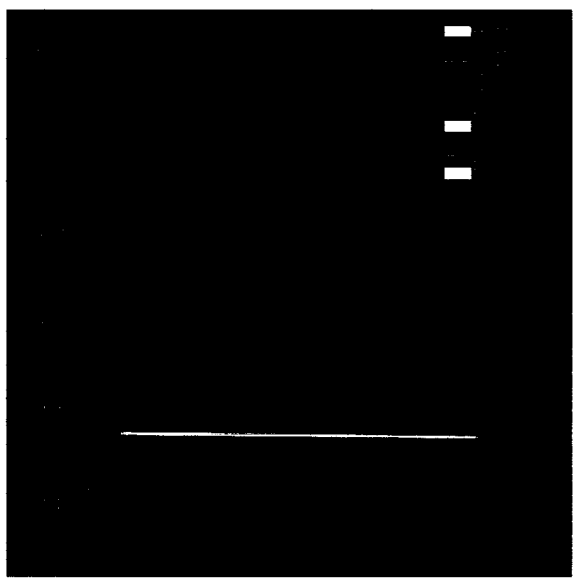


Figure 76: The solution diagram of AUTO demo `bvp`

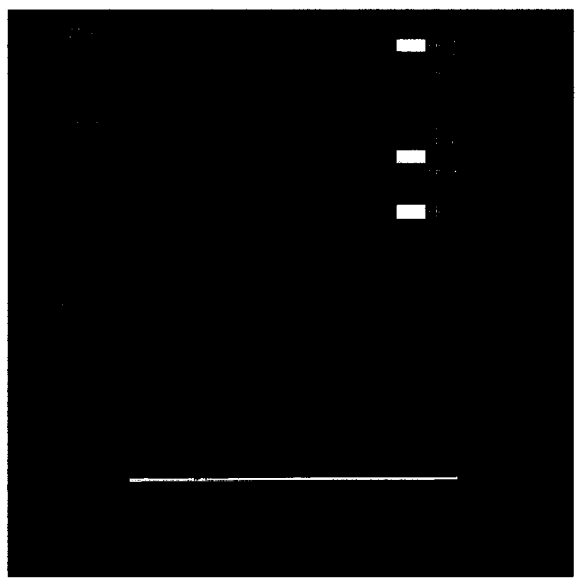


Figure 77: The bifurcation diagram of AUTO demo `bvp`

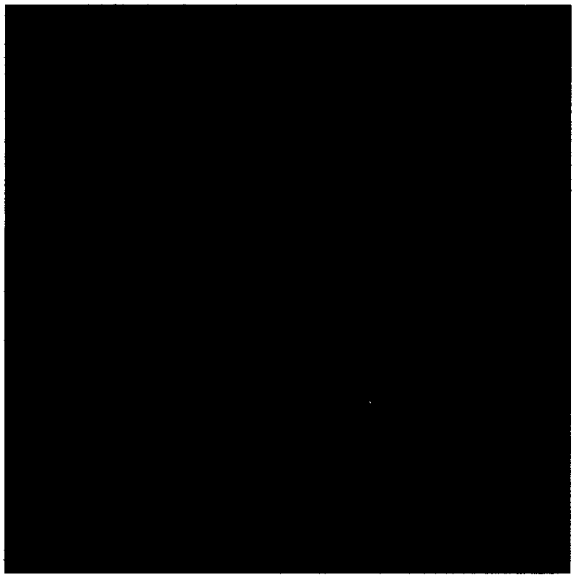


Figure 78: The bifurcation diagram of AUTO demo `brf`

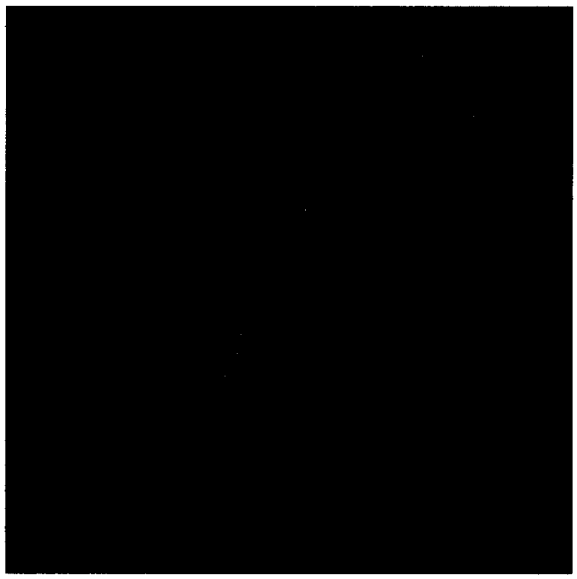


Figure 79: The bifurcation diagram of AUTO demo `bru`

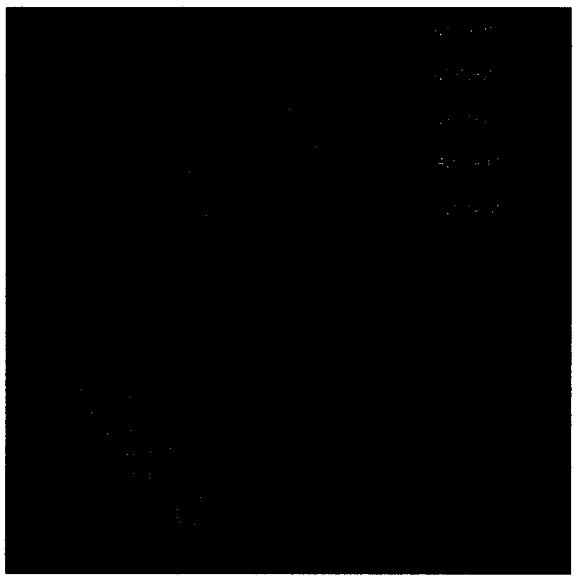


Figure 80: The solution diagram of AUTO demo `kpr`

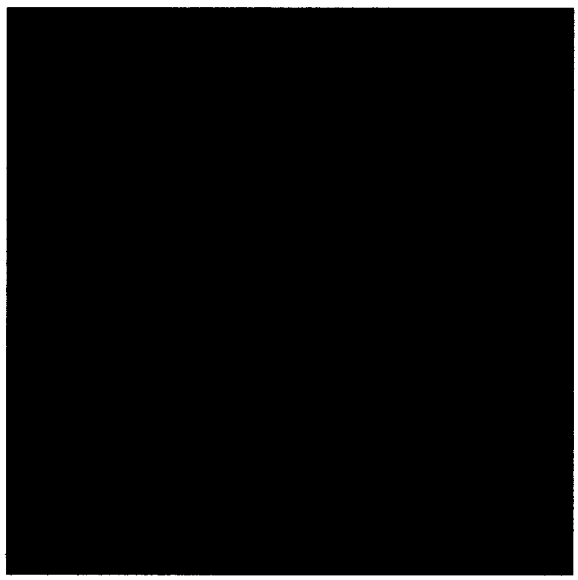


Figure 81: The solution diagram of AUTO demo `cir`

Chapter 6

Conclusions and Discussion

6.1 Conclusions

By introducing an unfolding parameter, AUTO has been successfully used for the computation of periodic solutions of the CR3BP, a conservative system.

Periodic solutions that emanate from the five libration points for the Sun-Earth, the Sun-Jupiter, and the Earth-Moon systems were computed using AUTO, and their bifurcation diagrams were depicted. The bifurcation diagrams give us a better understanding of how the periodic solutions are connected. This is of importance in the "interplanetary superhighway" research.

A new graphic data visualization package, PLAUT04, was developed for visualizing and analysing AUTO datasets. Features, such as an easy-to-use and easy-to-learn GUI, data animation, and flexible data rendering and coloring methods, make PLAUT04 a necessary tool for AUTO users. The specially designed animation feature for the CR3BP gives users an intuitive understanding of the motion of the three bodies in three dimensional space and in the inertial frame. We hope it will help users to have a better understanding of the CR3BP.

Data picking gives users a way to see the distribution of the Floquet multipliers. Zooming in/out gives users a way to view the details of the diagrams. 3D viewing allows users to observe the graphics from different directions.

The capability of rendering large datasets shows the robust design of PLAUT04. The speed of generating graphics from large datasets is reasonable.

Not only can PLAUT04 be used to create graphics from AUTO datasets for

CR3BP, but also it is a general purpose package for all applications of AUTO. Our examples have demontrated this.

6.2 Future development

PLAUT04 is based on Motif and Open Inventor. This makes it possible to run PLAUT04 on most Unix or Unix-like systems, without many changes. The current development was done under Linux. In the future, we will test the package under other Unix-like systems.

Some other development needs to be explored in the future. For example, the surface rendering in the current version is reasonably fast and in most cases the results are correct. However, it cannot generate a surface diagram from the data in some cases. In the future, the surface rendering method needs to be further optimized. In addition, more robust and efficient algorithms can be developed.

In the current version, there is little interoperation between the bifurcation diagram and the solution diagram. Because the bifurcation diagram and the solution diagram are closely related to each other, a future version of PLAUT04 should focus on the interoperability between them. This would make the data visualization even more intuitive for the users.

Bibliography

[1] E. L. Allgower and K. Georg. "Numerical Path Following." In P. G. Ciarlet and J. L. Lions, editors, Handbook of Numerical Analysis, Volume 5. North Holland Publishing, 1996.

[2] Charles R. Cowley. "Astronomy Exercise: An Exercise on Orbits and Orbital Histories." Retrieved September 9, 2004 from the World Wide Web: http://www.astro.lsa.umich.edu/users/cowley/R3B/

[3] U. M. Ascher, J. Christiansen, and R. D. Russell. "Collocation Software for Boundary Value ODEs." ACM Trans. Math. Software, 7, p209-222, 1981.

[4] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. "Numerical Solution of Boundary Value Problems for Ordinary Differential Equations." SIAM, 1995.

[5] U. M. Ascher and L. R. Petzold. "Computer Methods for ODEs and DAEs." SIAM, 1998.

[6] B. T. Barden and K. C. Howell. "Fundamental Motions near Collinear Libration Points and Their Transitions." J. Astronautical Sciences, 46(4), p361-378, 1998.

[7] W. -J. Beyn, A. Champneys, E. J. Doedel, W. Govaerts, B. Sandstede, and Yu. A. Kuznetov. "Numerical Continuation and Computation of Normal Forms." In B. Fiedler, editor, "Handbook of Dynamical Systems." Elsevier Science, 2, p149-219, 2001.

[8] J. -L. Basdevant and J. Dalibard. "The Two-Body Problem." Section 9.1 in "The Quantum Mechanics Solver: How to Apply Quantum Theory to Modern Physics." Berlin: Springer-Verlag, p61, 2000.

[9] Hinke Osinga. "Software for Dynamical Systems Theory." Retrieved September 9, 2004 from the World Wide Web:
http://www.enm.bris.ac.uk/staff/hinke/dss/

[10] J. Barrow-Green. "Poincaré and the Three Body Problem". Amer. Math. Soc., 1996.

[11] J. Henrard. "On Brown's Conjecture." Celestial Mechanics, 31, p115-122, 1983.

[12] J. V. Breakwell, and J. V. Brown. "The 'Halo' Family of 3-Dimensional Periodic Orbits in the Earth-Moon Restricted 3-Body Problem." Celestial Mechanics, 20, p289-404, 1979.

[13] E. W. Brown. "On the Oscillating Orbits about the Triangular Equilibrium Points in the Problem of the Three Bodies." Monthly Notices, Royal Astronomical Society 71, p438, 1911.

[14] Christian Marchal. "The Three-Body Problem." Elsevier, p258, 1990.

[15] Chenciner, A. and Montgomery, R. "A Remarkable Periodic Solution of the Three-Body Problem in the case of Equal Masses." Annals of Mathematics, 152, p881-901, 2000.

[16] A. C. Clarke. "Extra-terrestial Relays." Wireless World, p305-308, October 1945. Reprinted in Ascent to Orbit: A Scientific Autobiography, A.C. Clarke, Wiley, 1984.

[17] Neil J. Cornish. "The Lagrange Points." Retrieved September 9, 2004 from the World Wide Web:
http://www.physics.montana.edu/faculty/cornish/lagrange.html

[18] CMB Astrophysics Research Program. " COBE Information Page." Retrieved September 9, 2004 from the World Wide Web:
http://aether.lbl.gov/www/projects/cobe/COBE_home/cobe_home.html

[19] J. M. A. Danby. "Fundamentals of Celestial Mechanics." The MacMillan Company, 1962

[20] C. de Boor and B. Swartz. "Collocation at Gaussian Points." SIAM J. Numer. Anal., 10, p582-606, 1973.

[21] A. Deprit, and J. Henrard. "Construction of Orbits Asymptotic to a Periodic Orbit." Astron. J., 74, p308-316, 1969.

[22] D. J. Dichmann, E. J. Doedel, R. C. Paffenroth. "The Computation of Periodic Solutions of the 3-body Problem using the Numerical Continuation Software AUTO." in: "Libration Point Orbits and Applications". G. Gómez. M. W. Lo, J. J. Masdemont, eds., World Scientific, p489-528, 2003

[23] D. J. Dichmann, personal communication, 2003

[24] E. J. Doedel, *etc.*. AUTO Manual 2000.

[25] E. J. Doedel. "AUTO, a Program for the Automatic Bifurcation Analysis of Autonomous Systems." Congr. Numer., 30, p265-384, 1981.

[26] E. J. Doedel, H. B. Keller, and J. P. Kernévez. "Numerical Analysis and Control of Bifurcation Problems (I) Bifurcation in Finite Dimensions", International Journal of Bifurcation and Chaos, 1(3), p493-520, 1991.

[27] E. J. Doedel, H. B. Keller, and J. P. Kernévez. "Numerical Analysis and Control of Bifurcation Problems (II) Bifurcation in Infinite Dimensions", International Journal of Bifurcation and Chaos, 1(4), p745-772, 1991.

[28] E. J. Doedel. "Nonlinear Numerics." Int. J. Bifurcation and Chaos, 7(9), p2127-2143, 1997.

[29] E. J. Doedel. R. C. Paffenroth, H. B. Keller, D. J. Dichmann, J. Galán, and A. Vanderbauwhede. "Computation of Periodic Solutions of Conservative Systems with Application to the 3-Body Problem", International Journal of Bifurcation and Chaos, 13(6), p1-29, 2003.

[30] E. J. Doedel, R. C. Paffenroth, H. B. Keller, D. J. Dichmann, J. Galán, and A. Vanderbauwhede. "Elemental Periodic Orbits Associated with Libration Points in the Circular Restricted 3-Body Problem", (preprint), 2004.

[31] R. Farquhar, and A. K. Kamel. "Quasi-periodic Orbiters about the Translunar Libration Point." Celestial Mechanics, 7, p458-473, 1973.

[32] R. Farquhar. "The Control and Use of Libration-Point Satellites." PhD thesis, Department of Aeronautics and Astronautics, Stanford University, 1968.

[33] R. Farquhar. "The Flight of ISEEE-3/ICE: Origins, Mission History and a Legacy." J. Astronautical Science, 49(1):23-73, 2001.

[34] Fairgrieve and A. D. Jepson. "O. K. Floquet Multipliers." SIAM J. Numer. Anal., 28(5), p1446-1462, 1991.

[35] F. Felici, M. & F. Vanderbussche. "The ESA Astronomy Missions at L2: FIRST and Planck," J. Astronaut. Sci. 49, p185-196, 2001.

[36] D. Folta, S. Cookley, and K. Howell. "Trajectory Design Strategies for the NGST L2 Libration Point Mission." In AAS/AIAA Space Flight Mechanics Meeting, 2001. AAS 01-205.

[37] G. Gómez, J. Masdemont, and C. Simó. "Lissajous Orbits around Halo Orbits." In AAS/AIAA Space Flight Mechanics Meeting. 1997. AAS 97-106.

[38] G. Gómez, and N. J. Kasdin. "Optimal Out-of-the-ecliptic Trajectories for Space-borne Observatories." In AAS/AIAA Space Flight Mechanics Meeting, 2001, AAS 01-162.

[39] G. Gómez, J. Llibre, R. Martínez, and C. Simó. "Dynamics and Mission Design near Libration Points." Vol. II: Fundamentals: The Case of Triangular Libration Points, Vol. 3 (World Scientific, Singapore). 2001.

[40] G. Gómez, J. Llibre, R. Martínez, and C. Simó. "Dynamics and Mission Design near Libration Points." Vol. I: Fundamentals: The Case of Collinear Libration Points, Vol. 2 (World Scientific, Singapore). 2001.

[41] G. Gómez, J. Masdemont. "The Dynamics around the Collinear Equilibrium Points of the RTBP." Physica D, 157, p283-321, 2001.

[42] J. Hale, and H. Kocak. "Dynamics and Bifurcations." Springer-Verlag, 1991.

[43] Y. Hagihara. "Celestial Mechanics." Vol. 1 and 2, MIT Press, 1970-1972. Vol 3-5. Japan Society for the Promotion of Science, 1974-1976.

[44] M. Henderson. "Multiple Parameter Continuation: Computing Implicitly Defined k-Manifolds." Int. J. Bifurcation and Chaos, 12(3), p451-476, 2002.

[45] M. Hénon, J. M. Petit. 'Series Expansions for Encounter-type Solutions of Hill's Problem." Celestial Mechnics, 38, p67-100, 1986.

[46] M. Hénon. "Generating Families in the Restricted Problem." Springer-Verlag, 1997.

[47] K. C. Howell. "Three-dimensional, Periodic, 'Halo' Orbits." Celestial Mechanics, 32, p53-71, 1984.

[48] K. C. Howell, and E. T. Campbell. "Three-dimensional Periodic Solutions that Bifurcate from Halo Families in the Circular Restricted Three-Body Problem." In Spaceflight Mechanics, 1999. AAS 99-161

[49] K. C. Howell. "Families of Orbits in the Vicinity of the Collinear Libration Points." J. Astronautical Sciences, 49(1), p107-125, 2001.

[50] S. Ichtiaroglou, and M. Michalodimitrakis. "Three-Body Problem: The Existence of Families of Three-dimensional Periodic Orbits." Astronomy and Astrophysics, 81, p30-32, 1980.

[51] IEEE. " The First Information Visualization Symposium." IEEE Computer Society Press, 1995.

[52] A. D. Jepson. "Numerical Hopf Bifurcation." PhD thesis, Applied Mathematics, California Institute of Technology, 1981.

[53] H. B. Keller. "Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems." In P. H. Rabinowitz, editor, Applications of Bifurcation Theory, Academic Press, p359-384, 1977.

[54] Oksana Kotovych. "A Exactly Conservative Integrator for the N-Body Problem." Retrieved September 9, 2004 from the World Wide Web: http://www.math.ualberta.ca/ bowman/group/kotovych.pdf

[55] Kokhuitan. "The Story of the N-Body Problem." Retrieved September 9, 2004 from the World Wide Web:
http://members.fortunecity.com/kokhuitan/nbody.html

[56] Yuri A. Kuznetsov. "Elements of Applied Bifurcation Theory." Springer Verlag, 1998. Second Edition.

[57] K. Lust. "Improved Numerical Floquet multipliers." Int. J. Bifurcation and Chaos, 11(9), p2389-2410, 2001.

[58] European Space Agency. "Orbit/Navigation." Retrieved September 9, 2004 from the World Wide Web:
http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=34728

[59] M. W. Lo, B. Williams, W. Bollman, D. Han, Y. Hahn, J. Bell, E. Hirst, R. Corwin, P. Hong, K. Howell, B. Barden, and R. Wilson. "Genesis Mission Design". J. Astronaut. Sci. 49, p168-184, 2001.

[60] F. J. Muñoz-Almaraz, E. Freire, J. Galán, E. J. Doedel, and A. Vanderbauwhede. "Continuation of Periodic Orbits in Conservative and Hamiltonian Systems", Physica D, Vol. 181, No. 1-2, 2003, 1-38.

[61] J. R. Munkres. Analysis on Manifolds. Reading, MA: Addison-Wesley, 1991.

[62] B. H. McCormick, T. A. DeFanti, and M. D. Brown. "Visualization in Scientific Computing."

[63] F. R. Moulton. "An Introduction to Celestial Mechanics." (2nd Edition), New York: Macmillan, 1914.

[64] Marian Gidea, www.neiu.edu/~mgidea/perturbedsitnikov.pdf

[65] K. R. Meyer. "Periodic Solutions of the N-Body Problem." Berlin: Springer-Verlag, 1999.

[66] Minnesota Space Frontier Society. "The Earth-Moon System." Retrieved September 9, 2004 from the World Wide Web:
http://www.freemars.org/l5/aboutl5.html

[67] NASA. "The Wilkinson Microwave Anisotropy Probe. " Retrieved September 9, 2004 from the World Wide Web:
http://map.gsfc.nasa.gov/

[68] NASA. "WIND Spacecraft to Begin Petal Orbits" Retrieved September 9, 2004 from the World Wide Web:
http://pwg.gsfc.nasa.gov/istp/news/9804/

[69] NASA. "LOI: a 'Perfect Orbit Insertion'." Retrieved September 9, 2004 from the World Wide Web:
http://www.genesismission.org/mission/LOI.html

[70] NASA. "Wind." Retrieved September 9, 2004 from the World Wide Web:
http://www-istp.gsfc.nasa.gov/istp/wind/

[71] NASA. "ISTP Science News." Retrieved September 9, 2004 from the World Wide Web:
http://pwg.gsfc.nasa.gov/istp/news/9804/

[72] H. Poincaré. "Les Méthodes Nouvelles de la Mécanique Céleste." Gauthier-Villars. 1899

[73] L. Rosenblum *et al.* "Scientific Visualization Advances and Challenges." Harcourt Brace and Company, London, 1994.

[74] Archie E. Roy. "The Foundations of Astrodynamics." New York: Macmillan, 1967.

[75] W. C. Rheinboldt. "Numerical Analysis of Parametrized Nonlinear Equations." Wiley-Interscience, 1986. University of Arkansas Lecture Notes in the Mathematical Sciences.

[76] R. D. Russell and J. Christiansen. "Adaptive Mesh Selection Strategies for Solving Boundary Value Problems." SIAM J. Numer. Anal., 15, p59-80, 1978.

[77] Wikipedia. "Solar and Heliospheric Observatory." Retrieved September 9, 2004 from the World Wide Web:
http://en.wikipedia.org/wiki/Solar_and_Heliospheric_Observatory

[78] D. G. Saari. "A Visit to the Newtonian N-Body Problem via Elementary Complex Variables." Amer. Math. Monthly 97, p105-119, 1990.

[79] D. G. Saari, and Z. Xia. "Off to Infinity in Finite Time." Notices Amer. Math. Soc. 42, p538-546, 1995.

[80] R. Seydel. "From Equilibrium to Chaos. Practical Bifurcation and Stability Analysis (Second Edition)." Springer Verlag, New York, 1995.

[81] SGI. "Open Inventor." Retrieved September 9, 2004 from the World Wide Web: http://oss.sgi.com/projects/inventor/

[82] V. Szebehely. "Theory of Orbits. The Restricted Problem of Three Bodies". Academic Press, New York, 1967

[83] D. Taylor. "Horseshoe Periodic Orbits in the Restricted Problem of Three Bodies for a Sun-Jupiter Mass Ratio." Astronomy and Astrophysics 103, p288, 1981.

[84] F. Tisserand. "Traité de Mécanique Céleste", Tome 1, Gauthier Villars, 1960.

[85] C. W. Uphoff. "The Art and Science of Lunar Gravity Assist." Orbital Mechanics and Mission Design, vol 69 of Advances in the Astronautical Sciences, p333-346, 1989, AAS 89-170.

[86] Josie Wernecke. "The Inventor Mentor." Addison-Wesley, Boston, 1994.

[87] Eric W. Weisstein. "Restricted Three-Body Problem." Retrieved September 9, 2004 from the World Wide Web: http://scienceworld.wolfram.com/physics/RestrictedThree-BodyProblem.html

[88] G. G. Zagouras, P. G. Kazantzis. "Three-dimensional Periodic Oscillations Generating from Plane Periodic Ones around the Collinear Lagrangian Points," Astrophys. Space Sci. 61, p389-409, 1979.

Appendix A

How to Use AUTO for the CR3BP

AUTO is widely used for numerical bifurcation analysis. To learn more about using AUTO, see the AUTO reference manual [24]. Here we explain how to use AUTO to compute periodic solutions of the CR3BP.

A.1 The CR3BP equations

The equations, which model the CR3BP, are

$$\begin{aligned} x'' &= 2y' + x - (1-\mu)(x+\mu)/r_1^3 - \mu(x-1+\mu)/r_2^3 \ , \\ y'' &= -2x' + y - (1-\mu)y/r_1^3 - \mu y/r_2^3 \ , \\ z'' &= -(1-\mu)z/r_1^3 - \mu z/r_2^3 \ , \end{aligned} \tag{42}$$

where

$$\begin{aligned} r_1 &= \sqrt{(x+\mu)^2 + y^2 + z^2} \ , \\ r_2 &= \sqrt{(x-1+\mu)^2 + y^2 + z^2}. \end{aligned} \tag{43}$$

In order to use AUTO to continue periodic solutions of the CR3BP, we modify the original form of the equations as follows:

$$
\begin{aligned}
x' &= Tv_x \quad , \\
y' &= Tv_y \quad , \\
z' &= Tv_z \quad , \\
v_x' &= T[2v_y + x - \tfrac{(1-\mu)(x+\mu)}{r_1^3} - \tfrac{\mu(x-1+\mu)}{r_2^3}] + \lambda v_x \quad , \\
v_y' &= T[-2v_x + y - \tfrac{(1-\mu)y}{r_1^3} - \tfrac{\mu y}{r_2^3}] + \lambda v_y \quad , \\
v_z' &= T[\tfrac{-(1-\mu)z}{r_1^3} - \tfrac{\mu z}{r_2^3}] + \lambda v_z \quad .
\end{aligned}
\tag{44}
$$

Here λ is a scalar unknown that is solved for at each continuation step, its value will be zero (up to numerical precision) upon convergence of Newton's method. The periodicity boundary conditions, the integral phase condition, and the pseudo-arclength equation, as in Equation (27), are automatically added by AUTO, when computing periodic solutions. More details can be found in Chapter 2 of the thesis.

A.2 The CR3BP AUTO demo files

The CR3BP AUTO demo files can be found in the directory `auto/2000/demos/r3b/`. There are three sub-directories under it. The demo files for the Earth-Moon system are in the sub-directory `em`. The demo files for the Sun-Earth system are in the sub-directory `se`. The demo files for the Sun-Jupiter system are in the sub-directory `sj`. Each directory contains similar files. We take the Earth-Moon system as an example in this description.

The commands listed in Table 5 will copy the demo files to your work directory.

Table 6 lists all the files copied and their purpose.

Command	Action
mkdir cr3bp	make a directory for the CR3BP.
cd cr3bp	change the current directory to cr3bp.
@dm r3b/em	copy CR3BP demo files for the Earth-Moon system to the current directory. For the demo files of the Sun-Earth and Sun-Jupiter systems, you can use @dm r3b/se, and @dm r3b/sj, respectively.

Table 5: Copying the demo files for the Earth-Moon system

File	Purpose
r3b.c	the equations file for the CR3BP
c.r3b	the constants file
s.start	the start file for calculation for the Earth-Moon system
c.r3b.L1	the constants file for computing the **L1** family
c.r3b.H1	the constants file for computing the **H1** family

Table 6: The CR3BP demo files and their purpose

A.3 Listing of CR3BP demo files

Below is a listing of the equations file `r3b.c`. Function `func()` is defines the equations, and function `pvls()` is for user-defined parameters. Other functions are not used in our calculations.

```
/*=================================================================*/
/*=================================================================*/
/*===== The restricted 3-body problem: periodic solutions =======*/
/*=================================================================*/
/*=================================================================*/
#include "auto_f2c.h"
/*=================================================================*/
int func (integer ndim, const double *u, const integer *icp,
          const double *par, integer ijac, double *f, double *dfdu,
          double *dfdp)
```

```
{
        double x, y, z, xp, yp, zp, dE, dM, dE3, dM3, p, mu, mc;
int i;

        x  = u[0];      y  = u[1];      z  = u[2];
        xp = u[3];      yp = u[4];      zp = u[5];

        mu = par[1];
        p  = par[2];

        dE = sqrt((x+mu)*(x+mu) + y*y + z*z);
        dM = sqrt( (x-1+mu)*(x-1+mu) + y*y + z*z );
        mc = 1 - mu;
        dE3 = 1/(dE*dE*dE);
        dM3 = 1/(dM*dM*dM);

        f[0] = xp;
        f[1] = yp;
        f[2] = zp;
        f[3] =  2*yp + x - mc*dE3*(x+mu) - mu*dM3*(x-1+mu);
        f[4] = -2*xp + y - mc*dE3*y      - mu*dM3*y;
        f[5] =           - mc*dE3*z      - mu*dM3*z;

        f[3] += p*xp;
        f[4] += p*yp;
        f[5] += p*zp;

return 0;
}

/*==============================================================*/
/*==============================================================*/
int pvls (integer ndim, const double *u, double *par)
```

```
{
  extern double getp();
  double x, y, z, xp, yp, zp, d1, d2, mu, U, E;

        mu = par[1];
        x  = getp("BVO", 1, u);
        y  = getp("BVO", 2, u);
        z  = getp("BVO", 3, u);
        xp = getp("BVO", 4, u);
        yp = getp("BVO", 5, u);
        zp = getp("BVO", 6, u);

        d1 = sqrt((x+mu)*(x+mu) + y*y + z*z);
        d2 = sqrt( (x-1+mu)*(x-1+mu) + y*y + z*z );

        U = (x*x + y*y)/2 + (1-mu)/d1 + mu/d2;
        E = (xp*xp + yp*yp + zp*zp)/2 - U - mu*(1-mu)/2;
        par[3] = E;

        par[21] = getp("INT", 1, u);
        par[22] = getp("INT", 2, u);
        par[23] = getp("NRM", 3, u);

        return 0;
}
/*==================================================================*/
/*==================================================================*/
int stpnt() {}
int bcnd () {}
int icnd () {}
int fopt () {}
/*==================================================================*/
/*==================================================================*/
```

Table 7 shows the parameters defined in the CR3BP and their purpose.

Parameter	Purpose
1	the mass ratio μ
2	the unfolding parameter λ
3	the energy (Jacobi constant) E
10	the period T
21	$\int_0^1 x(t)\ dt$
22	$\int_0^1 y(t)\ dt$
23	$\sqrt{\int_0^1 z(t)^2\ dt}$

Table 7: AUTO parameters for the CR3BP

An AUTO constants file, named `c.r3b`, for the CR3BP is also provided. All constants used to control the calculations are set in this file. The best way to write a constants file is to use an old one as a template. An constants file for calculation of the **L**1 family is as follows:

```
6 2 11 0                NDIM,IPS,IRS,ILP
6 2 10     21 22 23 3   NICP,(ICP(I),I=1,NICP)
 50 4 3 3 -1 15 0 0     NTST,NCOL,IAD,ISP,ISW,IPLT,NBC,NINT
100 -1e9 1e9 -3 3       NMX,RL0,RL1,A0,A1
 5 -5 2 15 5 3 0        NPR,MXBF,IID,ITMX,ITNW,NWTN,JAC
1e-9 1e-9 1e-4          EPSL,EPSU,EPSS
 1e-2 1e-5 1e-1 1       DS,DSMIN,DSMAX,IADS
1                       NTHL,((I,THL(I)),I=1,NTHL)
10 0
0                       NTHU,((I,THU(I)),I=1,NTHU)
3                       NUZR,((I,UZR(I)),I=1,NUZR)
 10 1e2
 10 1e3
-10 1e4
```

The start point for our calculations is the solution with label 11 in the start file, `s.start`. In the start file, the label for each start point is comprised of a two-digit

number ij. The first digit i corresponds to the libration point Li, $i = 1, \cdots, 5$. The second digit j means that this is the jth family that emanates from Li. In our example, 11 corresponds to the "first" family that emanates from L1, namely, the family **L1**. If you want to compute the other families, you can select the label of the start point from Table 8.

Label	Libration Point	Family Reached
11	L1	**L1**
12	L1	**V1**
21	L2	**L2**
22	L2	**V2**
31	L3	**L3**
32	L3	**V3**
41	L4	**L4**
42	L4	**V4**
43	L4	**S4**

Table 8: Labels in the start file and their corresponding periodic families

A.4 Calculating the Lyapunov family L1

At this point, you can start the computations. Table 9 lists the commands for calculating the Lyapunov family **L1**.

Command	Action
cp c.r3b.1 c.r3b	get the first constants-file
@r r3b start	compute the Lyapunov family **L1**
@sv L1	save the output files as `b.L1, s.L1, d.L1`

Table 9: Calculating the Lyapunov **L1**

Execution of the commands will result in the following output to be printed on the screen.

```
$ @dm r3b/em
Copying demo r3b/em ... done
$ @r r3b start
gcc -O -DPTHREADS -O -I/usr/local/auto/2000/include -c r3b.c
gcc -O r3b.o -o r3b.exe /usr/local/auto/2000/lib/*.o  -lpthread -lpthread -L/usr/local/auto/2000/lib -lauto_f2c -lm
Starting r3b ...

  BR    PT  TY LAB    PAR(2)       L2-NORM U(3)    MAX(1)         PERIOD       PAR(21)       PAR(22)       PAR(23)       PAR(3)
  11     5      54  2.351461E-14  0.000000E+00  8.432422E-01  2.699025E+00  8.374743E-01 -1.402096E-15  0.000000E+00 -1.599117E+00
  11    10      55  6.283557E-15  0.000000E+00  8.521878E-01  2.730155E+00  8.397523E-01 -1.565030E-15  0.000000E+00 -1.594864E+00
  11    12  BP  56  2.385601E-15  0.000000E+00  8.548028E-01  2.743006E+00  8.406699E-01 -1.679280E-15  0.000000E+00 -1.593174E+00
  11    15      57  2.859613E-14  0.000000E+00  8.642796E-01  2.802097E+00  8.447246E-01 -1.903166E-15  0.000000E+00 -1.585860E+00
  11    20      58  1.186205E-13  0.000000E+00  8.847038E-01  2.970880E+00  8.549615E-01  8.947336E-16  0.000000E+00 -1.568430E+00
  11    25      59  1.607717E-12  0.000000E+00  9.153939E-01  3.273906E+00  8.693172E-01  1.846579E-13  0.000000E+00 -1.546023E+00
  11    30      60  1.500458E-11  0.000000E+00  9.549888E-01  3.774221E+00  8.851470E-01  5.313683E-12  0.000000E+00 -1.522652E+00
  11    32  BP  61  5.269546E-12  0.000000E+00  9.666750E-01  3.950048E+00  8.890273E-01  1.808889E-12  0.000000E+00 -1.516696E+00
  11    35      62 -1.941701E-11  0.000000E+00  1.002303E+00  4.573289E+00  8.975964E-01 -4.590192E-11  0.000000E+00 -1.500488E+00
 ... ...
  11    95      74 -1.711399E-07  0.000000E+00  1.196342E+00  7.184007E+00  4.719162E-01 -2.933416E-06  0.000000E+00 -1.194227E+00
  11   100  EP  75 -3.679044E-07  0.000000E+00  1.194461E+00  7.093447E+00  4.257231E-01 -5.555747E-06  0.000000E+00 -1.150220E+00

 Total Time    5.935E+00
r3b ... done
$ @sv L1
Saving fort.7 as b.L1 ... done
Saving fort.8 as s.L1 ... done
Saving fort.9 as d.L1 ... done
$
```

In this first run, we compute the Lyapunov family **L**1. Two branch points are located along **L**1. The first has label 56, and the second has label 61, as seen in the about output. The resulting data files are saved as "b.L1", "s.L1", and "d.L1".

Command	Action
cp c.r3b.H1 c.r3b	using the H1 constant file to follow the first branch point found in **L**1
@r r3b L1	compute **H**1
@sv H1	save the output files as b.H1, s.H1, and d.H1

Table 10: Calculating the Halo family **H**1 from **L**1

A.5 Calculating the Halo family from L1

We take the newly generated results `s.L1`, and `b.L1` as our start file, and we take the first branch point along it, namely label 56, as our restart point. To do this, we change IRS to 56 in the constant file `c.r3b`. Then we start the second run by following the commands in Table 10. Note that this time we use `@r r3b L1` to run

the computation, because now the start file is not `s.start`, but `s.L1`. The results are saved in "b.H1", "s.H1", and "d.H1".

Now you can use PLAUT04/r3b to view results of the above calculation. As to the use of PLAUT04, please refer to the PLAUT04 User's Guide.

A.6 Practical notes

In general, we set NTST to 50-100, EPSL, EPSU to 1.0e-9 or 1.0e-8, and DS, DSMIN, DSMAX, to -1e-2, 1e-6, 1e-1 respectively. However, some branches are difficult to compute, and we may have to use larger NTST, and smaller EPSL and EPSU, and adjust DS, DSMIN, and DSMAX.

Appendix B

User's Guide for Plaut04

Version 1.0

June, 2004

B.1 Product name

The name of this program is "PLAUT04", a graphic tool for AUTO data visualization. It includes utility tools, such as @rlb, @dlb, @klb, @ut *etc.*, described in Appendix C.

B.2 Document overview

This document covers the installation and use of PLAUT04. It also provides information for troubleshooting.

This user's guide contains all information an AUTO user needs, in order to view AUTO data sets with PLAUT04. An AUTO data set contains a solution file, "s.foo", a bifurcation file, "b.foo", and a diagnostic file, "d.foo". Here "foo" denotes a user-chosen data set name. This user's guide includes the following information:

1. The hardware and software requirements for running PLAUT04.

2. A description of how to install PLAUT04, and the configuration of the package.

3. A description of the PLAUT04 window system.

4. A list of PLAUT04 configuration options.

5. An example of using PLAUT04.

We assume that the reader has a basic knowledge of the AUTO package and of the Linux OS.

B.3 Requirements

Hardware

CPU: Pentium III 800 or higher.

Memory: 128 Megabytes or more.

Others :

- A graphics card is essential.
- At least 1 Megabyte hard disk space available for software installation.

Software

Operating System: - Red Hat Linux 7.3 or higher.

Applications:

- OpenGL or Mesa is essential,
- SGI Open Inventor 2.1.5.10, or Coin3D 2.2 with SoXt 1.1.0 and simage 1.6, must be present on the system.
- Motif 2.0 or higher is required.

B.4 Installation and configuration

B.4.1 Directory tree structure

The source tree containing the PLAUT04 source is structured as follows:

`data` - sample data

`doc` - reference manual pages

`src` - source files

`include` - headers

`widgets` - widgets used in the graphics

B.4.2 Dependencies

The source in its current form should compile on Red Hat Linux 7.3 or higher. Other dependencies include Open Inventor, X11R6, and Motif 2.0.

B.4.3 Building the tree

1. Make sure that the Open Inventor libraries are in the searching path.

2. The source code is compiled with optimization enabled by default.

3. Copy the source code tar ball, `plaut04.tar.z`, to `$AUTO_DIR`.

4. Change the current directory to `$AUTO_DIR` by typing:

   ```
   $ cd $AUTO_DIR
   ```

5. Extract the source code. The source code is packed using the Unix tar package. It can be extracted by issuing:

   ```
   $ tar xvfz plaut04.tar.z
   ```

   After this command has been executed, the source code will be in a newly created directory `$AUTO_DIR/plaut04/`.

6. Change the current directory to `plaut04` by typing:

   ```
   $ cd plaut04
   ```

   Now the working directory is `$AUTO_DIR/plaut04/`.

7. Configure the environment variables for compiling the source code. Before compiling the source code, the dependencies should be checked, and some environment variables need to be set. This can be done automatically by the configuration script:

   ```
   $ ./configure
   ```

8. Compile and install the source code. If the dependency check is passed, we can issue the following command to compile and install the package on the system.

   ```
   $ make
   ```

B.5 Setting up the resource file

The PLAUT04 resource file sets default values for almost all controls of PLAUT04. PLAUT04 allows us to write our own resources files and put them in the same directory as the AUTO data files. PLAUT04 first looks for the resource file in the current directory. If it cannot find a resource file there, then it will try to use the one installed in the AUTO root directory. If both these searches fail, then the internal default values will be used.

In order to write a usable resource file, one should follow the following rules:

1. Comment lines start with "#". Comments may take as many lines as desired.

2. Between the "variable name" and the default value, we must use "=" to tell the system that the left side is the "variable name", and the right side is its corresponding default value.

3. If a "variable" has aggregate values, a comma "," must be used between two values.

4. The line type is set using 4-digit hexadecimals, starting with "0x". Its values can range from 0 (invisible) to "0xffff" (solid). The system default is "0xffff" for stable solutions, and "0x3333" for unstable ones. The line pattern is determined by the number of 1s and 0s when the hexadecimal is converted to a 16-bit binary. A "1" indicates that the drawing occurs, and "0" that it does not, on a pixel by pixel basis. For example, the pattern "0xAAAA", in binary is 0000100010001000, and PLAUT04 interprets this as drawing 3 bits off, 1 bit on, 3 bits off, 1 bit on, 3 bits off, 1 bit on and finally 4 bits off. The pattern is read backward because the low order bits are used first.

5. Some variables can only be set to "Yes" or "No". They cannot be assigned other values.

6. No "variable name" should be modified.

It is strongly recommended that the default resource file is used as a template when writing a custom resource file.

Below is a copy of the default resource file.

```
#version 0.0

# Line colors are represented by RGB values from 0 to 1.0.
# DEFAULT color is also used when animationLabel == 0, i.e.,
# when showing all solutions and animating the solution change.
# Point Type    RED  GREEN  BLUE PATTERN
DEFAULT       = 1.0,  1.0,  1.0, 0xffff
BP            = 1.0,  0.0,  0.0, 0xffff
LP ALG        = 0.0,  1.0,  0.0, 0xffff
HB            = 0.0,  0.0,  1.0, 0xffff
UZ4           = 1.0,  1.0,  0.0, 0xffff
UZ-4          = 0.5,  0.5,  0.0, 0xffff
LP DIF        = 0.0,  0.0,  0.5, 0xffff
BP DIF        = 0.0,  0.5,  0.5, 0xffff
PD            = 1.0,  0.0,  1.0, 0xffff
TR            = 0.0,  1.0,  1.0, 0xffff
EP            = 0.3,  0.0,  0.3, 0xffff
MX            = 0.6,  0.0,  0.6, 0xffff
OTHERS        = 1.0,  1.0,  1.0, 0xffff

# Initialize the line pattern for showing stability
UNSTABLE LINE PATTERN = 0xffff
STABLE LINE PATTERN   = 0xffff

# Initialize the default options:
Draw Reference Plane  = No
Orbit Animation       = No
Satellite Animation   = No
Draw Primaries        = No
Draw Libration Points = No
Normalize Data        = Yes
Draw Background       = No

# Initialize the default coordinate axes:
#  0 --- None,
#  1 --- at origin
#  2 --- at left and behind
#  3 --- at left and ahead
Coordinate Type = 3
```

```
# Draw Scale on the Aexs
Draw Scale = Yes

#  Initialize the default graph type:
#  0 --- Solution (fort.8)
#  1 --- Bifurcation (fort.7)
Graph Type    = 0

# Initialize the default graph style:
#  0 --- LINES,
#  1 --- TUBES,
#  2 --- SURFACE
Graph Style  =  0

# Set the window width and height:
Window Width        = 1000
Window Height       = 1000

# Set X, Y, Z axes for the solution diagram:
# 0 is Time for X,Y,Z.
X Axis Solution             = 1
Y Axis Solution             = 2
Z Axis Solution             = 3

# Set X, Y, Z axes for the bifurcation diagram:
X Axis Bifurcation          = 4
Y Axis Bifurcation          = 5
Z Axis Bifurcation          = 6

#Labeled solutions:
Labels              = 0

# Set coloring method:
# -5 --- STABILITY
# -4 --- POINT
# -3 --- BRANCH
# -2 --- TYPE
# -1 --- LABEL
# Otherwise, according to the data in the ith column of the solution file.
# It can only be set to an integer value.
```

```
Coloring Method           = -2
Number of Period Animated = 1

# Line Width Scaler adjusts the thickness of curves:
Line Width Scaler         = 1.0

# The AniLine Thickness Scaler sets the thickness of animated solution curves:
AniLine Thickness Scaler  = 3.0

# Background color:
Background Color  = 0.0,  0.0,   0.0

# Background transparency:
Background Transparency = 0.0

# Disk transparency
# IF you turn Disk From File to "Yes", you should change the transparency there.
Disk Transparency = 0.7

# Read Disk From File
Disk From File = No

# Axes color:
X Axis Color    = 1.0,  0.0,   0.0
Y Axis Color    = 0.0,  1.0,   0.0
Z Axis Color    = 0.0,  0.0,   1.0

# Color of the satellite, large primary, and small primary in animation:
satellite Color             = 1.0,  0.0,   0.0
large primary Color         = 0.0,  1.0,   0.0
large primary tail Color    = 0.0,  1.0,   1.0
small primary Color         = 0.0,  0.0,   1.0
small primary tail Color    = 0.5,  0.5,   0.0

# Stable solution color:
Stable Solution Color = 0.0, 0.0, 1.0

# Stable solution color:
Unstable Solution Color = 1.0, 0.0, 0.0
```

```
# Set the radius of the satellite, large primary, and small primary:
# The normal size is 1.0.
# For smaller radius, use 0.xxx
# For bigger radius, use  X.XXX
Satellite Radius        = 1.0
Large Primary Radius    = 1.0
Small Primary Radius    = 1.0
Libration Point Size    = 1.0

# Set the maximum and minimum satellite animation speed:
Sat Max Animation Speed = 100
Sat Min Animation Speed = 0

# Set the maximum and minimum orbit-change animation speed:
Orbit Max Animation Speed = 100
Orbit Min Animation Speed = 0

# Set the active AUTO parameter indices:
parameter ID = 10

# Choose 3D or 2D graph:
3D  = Yes
```

B.6 Quick start

B.6.1 Starting and stopping Plaut04

Starting

The starting command for PLAUT04 is: "plaut04". A short AUTO97 command is also provided as "@pl". In the AUTO2000 Python CLUI, one can start PLAUT04 by typing "plaut04()" or "pl()".

This command can have no argument, one argument, or two arguments.

If no argument is provided, then the system uses the AUTO default data files, fort.7, fort.8, and fort.9, as inputs.

If one argument is given, it must be the name of the data set which we want to view. This data set should be in the current directory.

When two arguments are given, the first is always the path to the data set, and the second is the data set name.

Note that the AUTO data set name does not mean the full name of an AUTO file. It refers to the postfix of AUTO data files. For example, if we have the AUTO data files: "s.H1", "b.H1", and "d.H1", the AUTO data file name is "H1".

Stopping

One can exit the system by clicking the cross at the top-right corner of the window or from the "File" menu of the system.

B.6.2 Changing the "Type"

Often one will frequently change between the solution diagram and the bifurcation diagram. The "Type" menu helps to complete this change. This menu includes two items, "Solution", and "Bifurcation". There is a marker beside the current diagram. For example, if the current diagram is the solution diagram, but we want to change to the bifurcation diagram, we can do so by clicking "Type → Bifurcation" to switch to the bifurcation diagram.

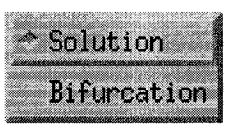


Figure 82: The Type Menu

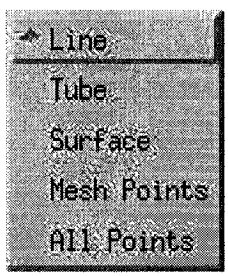


Figure 83: The Style Menu

B.6.3 Changing the "Style"

PLAUT04 provides four ways to draw the graphics, *i.e.*, using curves, tubes, points, or as a surface. One can select the style from the "Style" menu. The "Style" menu is shown in Figure 83.

B.6.4 Coordinate axes

Figure 84 shows the selections of the "Coord" menu. One may use this menu to select to show or not to show coordinate axes, and the type of coordinate axes, in the graphics.

B.6.5 Options

The "Options" Menu provides functions to add or remove widgets from the graphics. It also allows to start/stop solution or orbit animation. The "normalize data" normalizes the raw data to [0,1]. "Preference" lets us set preferences for the GUI (see Figure 85).

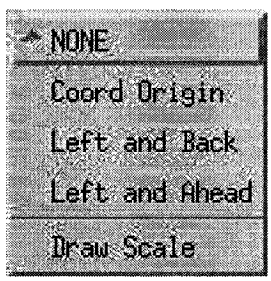


Figure 84: The Draw-Coordinate-Axes Menu

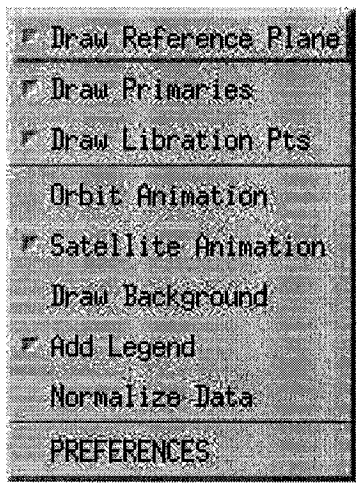


Figure 85: The Options Menu

B.6.6 CR3BP animation

The "Center" Menu allows to animate the motion of the three bodies in different coordinate systems. We can animate the motion in a large-primary-centered inertial coordinate system, or in a small-primary-centered inertial system, or in the barycentered inertial system. Figure 86 displays the layout of the "Center" menu.

B.6.7 Help

The "Help" menu provides an on-line help on how to use PLAUT04.

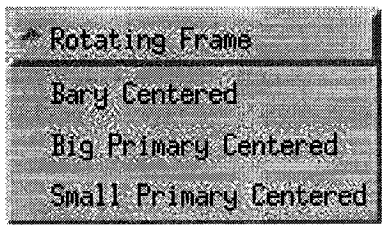


Figure 86: The Center Menu

Figure 87: The Help Menu

B.6.8 Picking a point in the diagram

The picking operation is useful when we want to know data corresponding to a certain point in the diagram. In order to execute a picking operation, we should follow these steps.

- Click the arrow icon to change the mouse to picking state.
- Move the mouse to the point of interest.
- Click the left button of the mouse to pick the point.

Once a point has been picked, a new window is popped up. In this new window, the Floquet multipliers of the point are shown in an x-y plane. Black crosses in the diagram indicate the Floquet Multipliers. The solution, and the values of the corresponding Floquet Multipliers, are given in the lower part of the window. A unit circle is drawn in the diagram. Figure 88 is an example of the picking operation. From this diagram, we can see that two Floquet Multipliers are outside the unit circle, two are on the unit circle, and the other two are inside the unit circle.

B.6.9 Choosing the variables

AUTO can generate large amounts of data. The CR3BP, for example, has 6 variables, *i.e.*, x, y, z, x', y', z', and time. One can choose to draw any combination of these variables in 2 or 3 dimensions using PLAUT04. On the list bar, we can see three dropdown lists with label "X", "Y", and "Z" (See Figure 89). Each of these three lists has the exact number of choices, namely, the number of variables of the system plus one. In our case, these lists have 7 choices, which are represented by the integers 0 to 6. 0 represents time. 1 to 6 stand for x, y, z, x', y', and z', respectively. "1" is selected for "X", which indicates that x is drawn on the X-axis. "2" is selected for

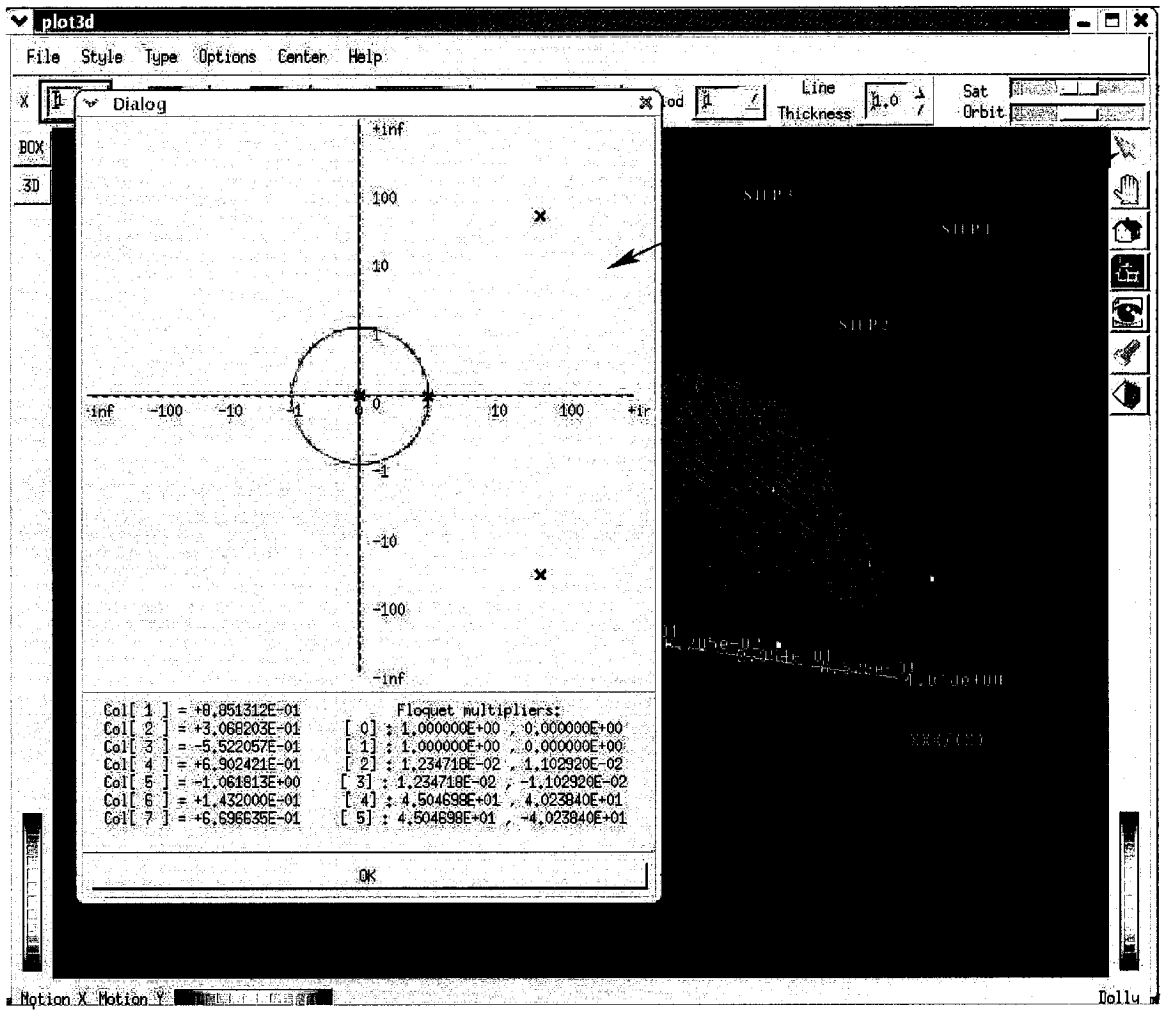


Figure 88: Picking a point

"Y", which indicates that y is represented on the Y-axis. "3" is selected for "Z", which indicates that z is represented on the Z-axis.

We can also show multiple combinations at the same time. For example, if we want to show x-y-z and x'-y'-z' in the same diagram, we can input $1, 4$ in the "X" dropdown list to select x and x' being drawn on the X-axis, input $2, 5$ in the "Y" list to show y and y' on the Y-axis, and input $3, 6$ in the "Z" dropdown list to draw z and z' on the Z-Axis. Note that after finishing the input in the dropdown list box, we must type "ENTER" for the input to be accepted by the system. Figure 90 shows the results of the above choices. The combination is flexible. For example, if X is 1, Y is $3, 5$, and Z is $4, 5, 6$, the system will automatically reorganize them to $1 - 3 - 4$,

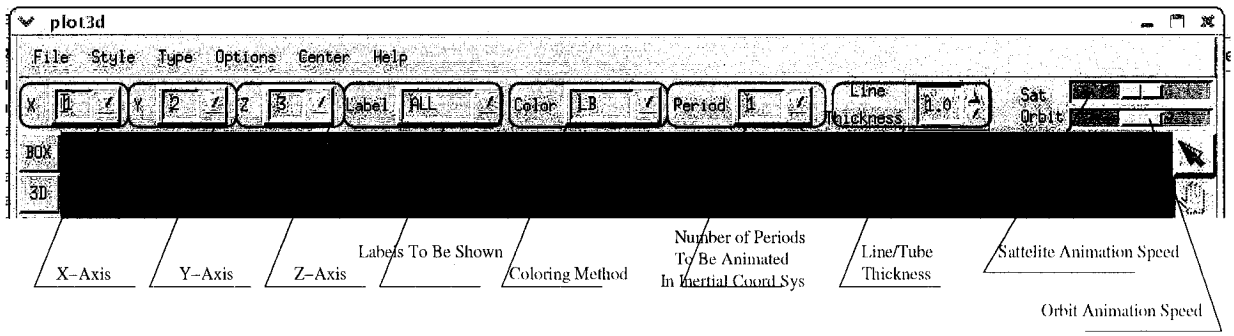


Figure 89: Menu-bar layout

$1-5-5$, $1-3-6$ and show the results. If X is $1, 5$, Y is 2, and Z is $3, 4$, the system reorganizes them to $1-2-3$, $5-2-4$.

Different components are drawn with different colors from blue to red.

The default values can be set in the resource file. If no resource file exists, then the system will use "1" for X-axis, "2" for Y-axis, and "3" for Z-axis for both the solution and the bifurcation diagrams.

B.6.10 Choosing labels

From the Label list, we can choose the label of the solution to be drawn. If "ALL" is chosen, all solutions are shown in the diagram. If "NONE" is chosen, none of the solutions is shown. "HALF" shows the solutions with odd labels and special solutions only. "SPEC" lets the system show the special solutions only. We can also show selected solutions by inputting their labels in the list box separated by commas. For example, typing 1, 10, 15, 20 will lead the system to show only the solutions with label 1, 10, 15 and 20.

We can set the default value for this list in the PLAUT04 resource file.

B.6.11 Coloring

Many coloring methods are provided. They can be classified into three groups. The first group is coloring by variables. This group provides as many choices as the number of variables of a problem plus 1 for the time. The second group is coloring by parameters. These parameters are defined by the AUTO user. in the AUTO constants file. There are as many choices as the number of parameters defined in the

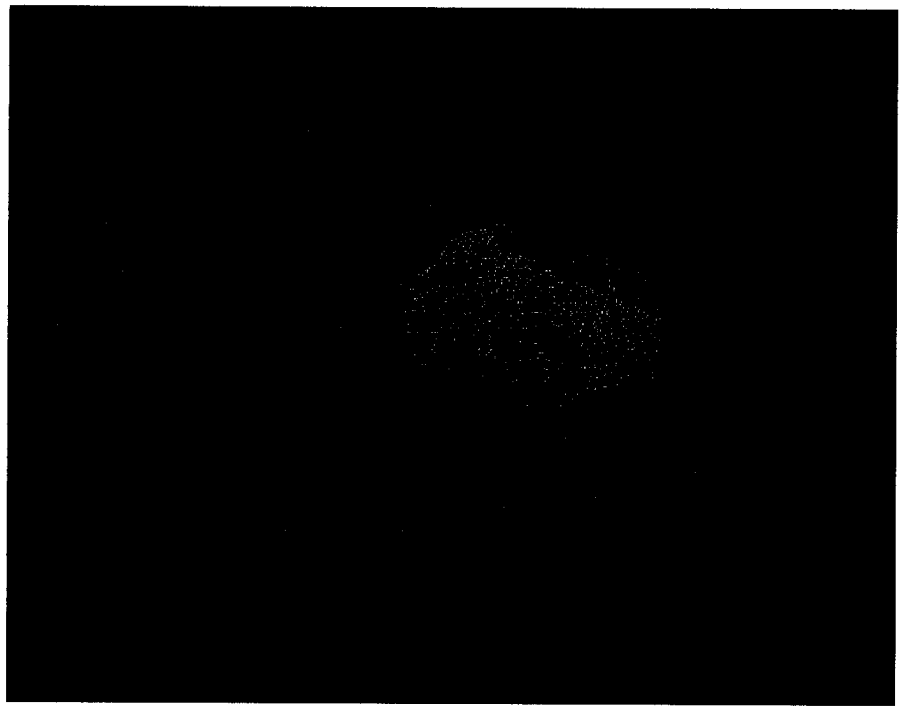


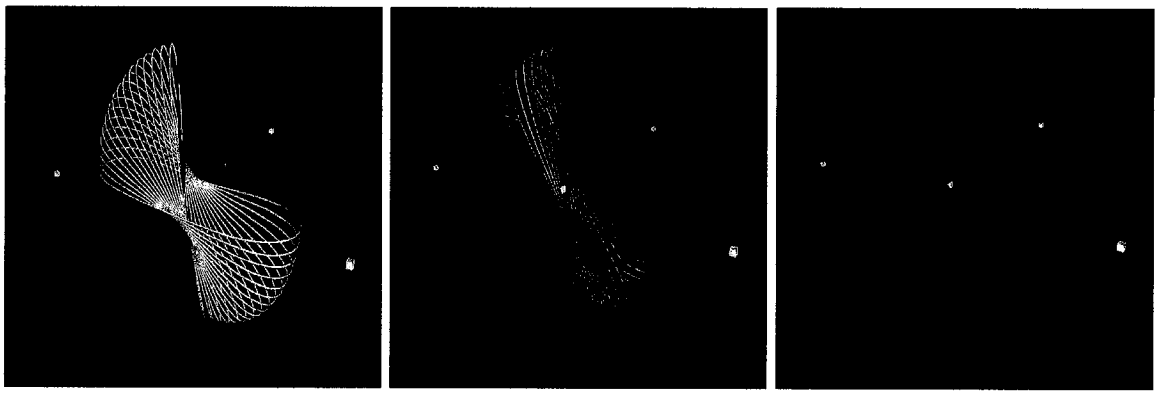
Figure 90: Displaying multiple components

AUTO constants file. The third group includes "TYPE", type of solution, "PONT", point number, "BRAN", the branch to which the solution belongs, and "LABL", label of the solution. Different coloring methods cannot be used at the same time. Figure 91 shows the difference between coloring by type and coloring by label. From Figure 91(a), we can see that there is only one branching orbit in this family, which is shown in cyan. In Figure 91(b), the start solution is colored in blue, and the last solution is colored in red. When using time to color the diagram, 0 is set to blue, while 1 is set to red.

We can set the default value in the PLAUT04 resource file.

B.6.12 Number of periods to be animated

Generally only one period is animated when we animate the solution in the inertial frame. However, the SpinBox allows us to change the default value. This is a specially designed function for the CR3BP. It is useful when we animate the motion in the three bodies in the inertial frame.



(a) Coloring by "Type" (b) Coloring by "Label" (c) Coloring by "Time"

Figure 91: Coloring

B.6.13 Changing the line/tube thickness

The "Line Thickness" spinbox allows us to increase or decrease the line/tube thickness in the diagram. The PLAUT04 resource file also provides a way to change the default values of the line/tube thickness.

B.6.14 Changing the animation speed

The "Sat" and "Orbit" scale bar allow us to change the animation speed. Their Maximum and Minimum value can be set in the resource file.

B.6.15 Changing the background picture

A user can set the backgroud with his favorite picture. To do this, a user should copy the picture to the directory "$AUTO_DIR/plaut04/widgets", and then change the name of the file to "background.rgb".

B.7 Example

In this example, we want to view a CR3BP data set. We want the diagram to show the "x" component on the X-axis, "y" component on the Y-axis, and "z" component on the Z-axis for the solution diagram. In the CR3BP, we use the parameters "1 2 3

10 21 22 23" in the AUTO calculations, and we also want to be able to use these to color the diagram, so we set the "parameter indices".

Other preferences include

- The diagram is drawn using Tubes.
- Coordinate axes are not drawn.
- No animation.
- Reference plane, libration points, and primaries are drawn.
- All labels are shown.
- Data is not normalized.

The settings are the settings in the resource file are then as follows:

```
#  Initialize the default options
Draw Reference Plane  = Yes
Orbit Animation       = No
Satellite Animation   = No
Draw Primaries        = Yes
Draw Libration Points = Yes
Normalize Data        = No
Draw Background       = No

#  Initialize the default graph type
#  0 --- Solution(fort.8)  1 --- Bifurcation(fort.7)
Graph Type    = 0

# initialize the default graph style
#  0 --- LINES,  1 --- TUBES, 2 ---- SURFACE  3--- nurbs curve
graph Style  =  1

# set X, Y, Z, and Label
# 0 is Time for X,Y,Z. 0 is "All" for Label

Solution X Axis              = 1
Solution Y Axis              = 2
Solution Z Axis              = 3
```

```
Labels                          = 0

#set the parameter indices
parameter ID = 1,  2,  3, 10,  15, 21, 22,  23
```

Based on the above settings, the solution diagram for the CR3BP family L1 for $\mu = 0.01215$ appears in Figure 92.

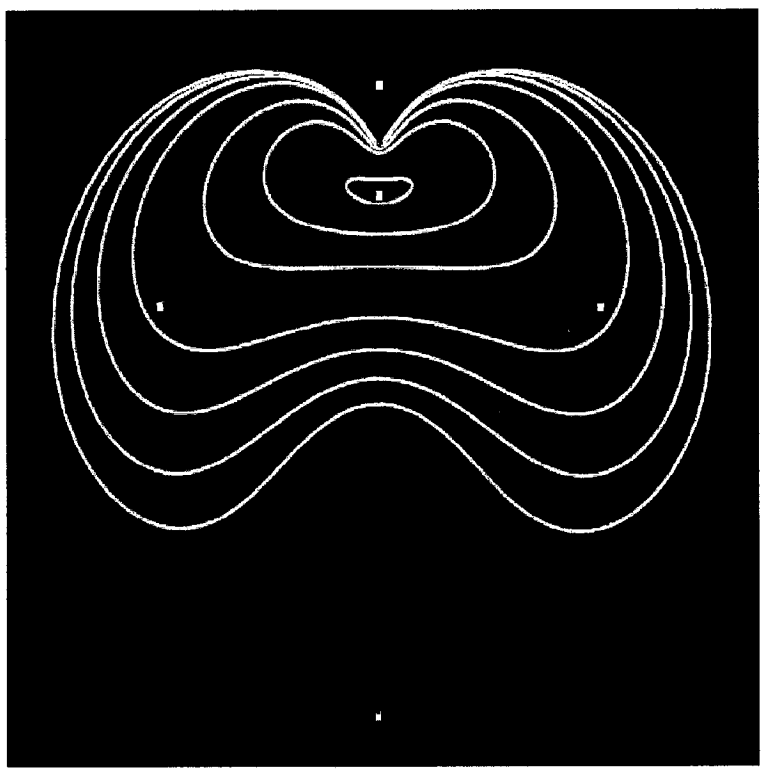


Figure 92: Example

B.8 Miscellaneous

1. Where can I download the Open Inventor libraries?

   One can download SGI's implementation from:

   `ftp://oss.sgi.com/projects/inventor/download/`

   Because SGI's implementation for Linux cannot show text correctly, we recommend that Coin be used instead of SGI's implementation. Coin3D can be downloaded from:

   `http://www.coin3d.org/download/`

2. Why does the system tell me: " plaut04: error while loading shared libraries: libCoin.so.40: cannot open shared object file: No such file or directory"?

This message means that the system cannot find the Coin libraries. To solve this problem, make sure that the Coin libraries are reachable, namely, that the environment variable LD_LLIBRARY_PATH is set to the directory where the Coin libraries are installed.

For example, if Coin3D is installed in `''/usr/local/lib''`, in order to make PLAUT04 work properly, the following code should be added in the `.bash_profile`:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib:/usr/local/lib
```

Appendix C

AUTO Utilities

Some additional AUTO utilities were written during the development of PLAUT04. In this appendix, we give a brief introduction to their function and we demonstrate their use. We use the abbreviation "label" to denote "labeled solution".

C.1 New AUTO97 commands

C.1.1 @ut command

Purpose

Does maintenance operations on the AUTO data files "s.foo" and "b.foo". One can use this command to display labels, delete labels, relabel, *etc.*

Format

```
@ut [source_filename [back_filename]]
```

Description

Type `@ut` to maintain "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".

Type `@ut foo` to maintain "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".

Type `@ut foo bar` to maintain "s.foo" and "b.foo". The original files are backed up as "s.bar~" and "b.bar~".

This tool has an interactive command interface. The available commands are:

- h print system help information.
- l list the labels in the solution files.
- d delete labels in the solution files.
- u recover the deleted labels in the solutions before saving them. If the changes have been written to the disk, this command does not work.
- r relabel solutions.
- k keep certain solutions.

The above commands can have arguments with it. We can provide the arguments in the following ways:

Command format	Example	Description
command [labels]	l 12	list label 12
	l 2-5 10 12	list labels from 2 to 5, 10, and 12.
command [ty (type of the labels)]	l ty bp ep	list all labels of type "bp", and "ep"
command [ty= (type of the labels)]	l ty= bp ep	list all labels of type "bp", and "ep"

If no argument is provided, the objects are all labels in the file.

Note that there always must be a space after "ty=" or "ty".

See also

@rdc, @llb, @klb, @dlb, @rlb.

C.1.2 @rdc command

Purpose

Reduce the size of "s.foo" or "fort.8" by removing every other user defined points from the solution files. However, if a label is a special point, it is kept.

Format

```
@rdc [filename [back_filename]]
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename". The original files are backed up as "s.filename~" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, the original files are backed up as "fort.7~" and "fort.8~". If we want to back up a file using a different name, it can be done by the command `@rdc source backfile`.

Examples

`@rdc`	reduce "fort.8". The original files are backed up as "fort.8~" and "fort.7~".
`@rdc foo`	reduce "s.foo". The original files are backed up as "s.foo~" and "b.foo~".
`@rdc foo bar`	reduce "s.foo", the original files are backed up as "s.bar~" and "b.bar~".

See also

@ut.

C.1.3 @rlb command

Purpose

Relabels the solutions in the solution files.

Format

```
@rlb [[s=|solution=] filename] [labels] |
     [ty=|tp=|type=|ty|tp|type (type of labels)]
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename~" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and the "types of the labels" should not be provided at the same time. If we want to use the "type of labels" option, the keywords "ty" or "ty=" must be used before choosing the types.

Examples

`@rlb`	relabel all solutions in "fort.7" and "fort.8". The new labels are generated automatically from 1 to m. The original files are backed up as "fort.7~" and "fort.8~".
`@rlb 2, 5-10 15 17 20-23`	give new labels to label 2, labels from 5 to 10, 15, 17, and labels from 20 to 23, in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".
`@rlb ty= bp`	give new labels to the labels of type "bp" in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".
`@rlb foo`	give new labels to all labels in "s.foo" and "b.foo". The new labels are generated automatically from 1 to m. The original files are backed up as "s.foo~" and "b.foo~".
`@rlb foo 1 5`	give new labels to the current label 1 and 5 in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`@rlb foo ty= lp hb tr`	give new labels to the labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo". The system asks for new labels for each of the points, after we type the "ENTER" key. The original files are backed up as "s.foo~" and "b.foo~".
`@rlb s= foo ty= lp hb tr`	the same as `@rlb foo ty= lp hb tr`.

See also

@llb, @klb, @dlb, @ut.

C.1.4 @llb command

Purpose

List solution labels.

Format

```
@llb [[s=|solution=] filename] [labels] |
     [ty=|tp=|type=|ty|tp|type (type of labels)]
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`@llb`	list all labels in "fort.7" and "fort.8".
`@llb 2, 5-10 15 17 20-23`	list label 2, labels 5 to 10, 15, 17, and labels 20 to 23, in "fort.7" and "fort.8".
`@llb ty= bp`	list labels of type "bp" in "fort.7" and "fort.8".
`@llb foo`	list all labels in "s.foo" and "b.foo".
`@llb foo 1 5`	list label 1 and 5 in "s.foo" and "b.foo".
`@llb foo ty= lp hb tr`	list labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo".
`@llb s= foo ty lp hb tr`	the same as `@llb foo ty= lp hb tr`.

See also

@klb, @rlb, @dlb, @ut.

C.1.5 @klb command

Purpose

Keep selected labels, or labels of designated type in the solution files, while deleting all other labels.

Format

```
@klb [[s=|solution=] filename] [labels] |
     [ty=|tp=|type=|ty|tp|type (type of labels)]
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`@klb`	keep all labels in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".
`@klb 2, 5-10 15 17 20-23`	keep the given labels, *i.e.*, label 2, labels 5 to 10, 15, 17, and labels 20 to 23, in "fort.7" and "fort.8". All other labels are deleted. The original files are backed up as "fort.7~" and "fort.8~".
`@klb ty= bp`	keep labels of type "bp" in "fort.7" and "fort.8". All labels with other types are deleted. The original files are backed up as "fort.7~" and "fort.8~".
`@klb foo`	keep all labels in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`@klb foo 1 5`	keep the solutions with label 1 and 5 in "s.foo" and "b.foo". Other solutions are deleted. The original files are backed up as "s.foo~" and "b.foo~".

`@klb foo ty= lp hb tr`	keep labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo". Labels with other types are deleted. The original files are backed up as "s.foo~" and "b.foo~".
`@klb s= foo ty= lp hb tr`	the same as `@klb foo ty= lp hb tr`.

See also

@llb, @dlb, @rlb, @ut.

C.1.6 @dlb command

Purpose

Delete the selected labels or labels with the designated type in the solutions.

Format

```
@dlb [[s=|solution=] filename] [labels] |
     [ty=|tp=|type=|ty|tp|type (type of labels)]
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename~" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`@dlb`	delete all labels in "fort.7" and "fort.8". In order to prevent a user from deleting everything in the files in case of carelessness, the system asks the user to confirm each deletion one by one. The original files are backed up as "fort.7~" and "fort.8~".

`@dlb 2, 5-10 15 17 20-23`	delete the given labels, *i.e.*, label 2, labels 5 to 10, label 15, 17, and labels 20 to 23, in "fort.7" and "fort.8". Other labels are kept. The original files are backed up as "fort.7~" and "fort.8~".
`@dlb ty= bp`	delete labels of type "bp" in "fort.7" and "fort.8". All other types are kept. The original files are backed up as "fort.7~" and "fort.8~".
`@dlb foo`	delete all labels in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`@dlb foo 1 5`	delete label 1 and 5 in the files "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`@dlb foo ty= lp hb tr`	delete labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo ".
`@dlb s= foo ty= lp hb tr`	the same as `@dlb foo ty= lp hb tr`.

See also

@llb, @klb, @rlb, @ut.

C.2 New AUTO2000 commands

C.2.1 ut() command

Purpose

Does maintenance operations on the AUTO data files "s.foo" and "b.foo". One can use this command to display labels, delete labels, relabel, *etc.*

Format

```
ut([solution_file_name,[backup_file_name]])
```

Description

Type `ut()` to maintain "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".

Type `ut('foo')` to maintain "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".

Type `ut('foo', 'bar')` to maintain "s.foo" and "b.foo". The original files are backed up as "s.bar~" and "b.bar~".

This tool has an interactive command interface. The available commands for it are:

h	print system help information.
l	list the labels in the solution file.
d	delete labels in the solution file.
u	recover the deleted labels in the solutions before saving them. If the changes have been written to the disk, this command does not work.
r	relabel solutions.
k	keep certain solutions.

The above commands can have arguments with it. We can provide the arguments in the following approaches:

Command format	Example	Description
command [labels]	l 12	list label 12
	l 2-5 10 12	list labels from 2 to 5, 10, and 12.
command [ty (type of the labels)]	l ty bp ep	list all labels of type "bp", and "ep"
command [ty= (type of the labels)]	l ty= bp ep	list all labels of type "bp", and "ep"

If no argument is provided, the objects are all labels in the files.

Note that there must always be a space after "ty=" or "ty".

Aliases

ut utility.

See also

rdc, llb, klb, dlb, rlb.

C.2.2 rdc() command

Purpose

Reduce the size of "s.foo" or "fort.8" by removing every other user defined points from the solution file. However, if a label is a special point, it is kept.

Format

`rdc([solution_file_name,[backup_file_name]])`

Description

If the 's.source_filename' is given, the operation objects are "s.filename" and "b.filename". The original files are backed up as "s.filename~" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~". If we want to back up a file using a different name, it can be done by the command `rdc(source, backfile)`.

Examples

`rdc()`	reduce "fort.8". The original files are backed up as "fort.8~" and "fort.7~".
`rdc('foo')`	reduce "s.foo". The original files are backed up as "s.foo~" and "b.foo~".
`rdc('foo', 'bar')`	reduce "s.foo", the original files are backed up as "s.bar~" and "b.bar~".

Aliases

reducelabel reduce rdc.

See also

ut.

C.2.3 llb() command

Purpose

List labels in the solution/bifurcation files.

Format

```
llb([[s=|solution=]solution_file_name] ,
    [[t=|ty=|type=]type_of_labels]      |
    [[l=|lb=|label=]labels])
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`llb()`	list all labels in "fort.7" and "fort.8".
`llb(lb='2, 5-10 15 17 20-23')`	list label 2, labels from 5 to 10, 15, 17,and labels from 20 to 23, in "fort.7" and "fort.8".
`llb(ty= 'bp')`	list labels of type "bp" in "fort.7" and "fort.8".
`llb('foo')`	list all labels in "s.foo" and "b.foo".
`llb('foo', l='1 5')`	list label 1 and 5 in "s.foo" and "b.foo".
`llb('foo', ty='lp hb tr')`	list labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo".
`llb(s= 'foo', ty='lp hb tr')`	the same as `llb('foo', ty='lp hb tr')`.

Aliases

llb listlabel.

See also

klb, dlb, rlb, ut.

C.2.4 klb() command

Purpose

Keep selected labels, or labels of designated type, in the solution files, while deleting all other labels.

Format

```
klb([[s=|solution=]solution_file_name] ,
    [[t=|ty=|type=]type_of_labels]      |
    [[l=|lb=|label=]labels])
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`klb()`	keep all labels in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".
`klb(lb='2, 5-10 15 17 20-23')`	keep the given labels, *i.e.*, label 2, labels 5 to 10, 15, 17, and labels 20 to 23, in "fort.7" and "fort.8". All other labels are deleted. The original files are backed up as "fort.7~" and "fort.8~".
`klb(ty= 'bp')`	keep labels of type "bp" in "fort.7" and "fort.8". All labels with other types are deleted. The original files are backed up as "fort.7~" and "fort.8~".
`klb('foo')`	keep all labels in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`klb('foo', l='1 5')`	keep the solutions with label 1 and 5 in "s.foo" and "b.foo". All other solutions are deleted. The original files are backed up as "s.foo~" and "b.foo~".
`klb('foo', ty= 'lp hb tr')`	keep labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo". Labels with other types are deleted. The original files are backed up as "s.foo~" and "b.foo~".
`klb(s= 'foo', ty= 'lp hb tr')`	the same as `klb('foo', ty='lp hb tr')`.

Aliases

klb keeplabel.

See also

llb, dlb, rlb, ut.

C.2.5 rlb() command

Purpose

Relabel the solution/bifurcation files.

Format

```
rlb([[s=|solution=]solution_file_name] ,
    [[t=|ty=|type=]type_of_labels]      |
    [[l=|lb=|label=]labels])
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename~" and "b.filename~". Otherwise, "fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and the "types of the labels" should not be provided at the same time. If we want to use the "type of the labels" option, the keywords "ty" or "ty=" must be used before choosing the types.

Examples

`rlb()`	relabel all solutions in "fort.7" and "fort.8". The new labels are generated automatically from 1 to m. The original files are backed up as "fort.7~" and "fort.8~".
`rlb(lb='2, 5-10 15 17 20-23')`	give new labels to label 2, labels from 5 to 10, 15, 17, and labels from 20 to 23, in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".
`rlb(ty= 'bp')`	give new labels to the solutions of type "bp" in "fort.7" and "fort.8". The original files are backed up as "fort.7~" and "fort.8~".

`rlb('foo')`	give new labels to all solutions in "s.foo" and "b.foo". The new labels are generated automatically from 1 to m. The original files are backed up as "s.foo~" and "b.foo~".
`rlb('foo',l='1 5')`	give new labels to the current label 1 and 5 in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`rlb('foo', ty='lp hb tr')`	give new labels to solutions with type "lp", "hb", and "tr" in "s.foo" and "b.foo". The system asks for new labels for each of the points, after we type the "ENTER" key. The original files are backed up as "s.foo~" and "b.foo~".
`rlb(s= 'foo', ty= 'lp hb tr')`	the same as `@rlb foo ty= lp hb tr`.

Aliases

relabel rlb.

See also

llb, klb, dlb, ut.

C.2.6 dlb() command

Purpose

delete labels in the solution/bifurcation files.

Format

```
dlb([[s=|solution=]solution_file_name] ,
    [[t=|ty=|type=]type_of_labels]      |
    [[l=|lb=|label=]labels])
```

Description

If the "filename" is given, the operation objects are "s.filename" and "b.filename", and the original files are backed up as "s.filename~" and "b.filename~". Otherwise,

"fort.7" and "fort.8" are used, and the original files are backed up as "fort.7~" and "fort.8~".

The options "labels" and "types of the labels" should not be used at the same time.

Examples

`dlb()`	delete all labels in "fort.7" and "fort.8". In order to prevent a user from deleting everything in the files in case of carelessness, the system asks the user to confirm each deletion one by one. The original files are backed up as "fort.7~" and "fort.8~".
`dlb(lb='2, 5-10 15 17 20-23')`	delete the given labels, *i.e.*, label 2, labels 5 to 10, label 15, 17, and labels 20 to 23, in "fort.7" and "fort.8". All the other labels are kept. The original files are backed up as "fort.7~" and "fort.8~".
`dlb(ty= 'bp')`	delete labels of type "bp" in "fort.7" and "fort.8". All other types are kept. The original files are backed up as "fort.7~" and "fort.8~".
`dlb('foo')`	delete all labels in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`dlb('foo',l='1 5')`	delete label 1 and 5 in the files "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo~".
`dlb('foo', ty= 'lp hb tr')`	delete labels with type "lp", "hb", and "tr" in "s.foo" and "b.foo". The original files are backed up as "s.foo~" and "b.foo "
`dlb(s= 'foo', ty= 'lp hb tr')`	the same as `dlb('foo', ty='lp hb tr')`.

Aliases

dlb deletelabel.

See also

llb, rlb, klb, ut.