UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

El problema de tres cossos i bicircular

Autor: Arnau Ruiz Sebastián

Director: Dr. Àngel Jorba
Realitzat a: Departament de Matemàtiques aplicades

Barcelona, 9 de juny de 2024

Abstract

The main goal in this work is to study the restricted three-body problem and the bicircular model, along with the research of several periodic orbits in the bicircular model. From the restricted three-body problem we will treat the deduction of the equation of motion in the sinodic coordinates, the curves of zero velocity, the equilibrium points and their lineal stability. On the other hand, we will only deduce the equation of motion of the bicircular model. Afterwards, we will define several numeric algorithms in order to end the text searching for periodic orbits in the bicircular model.

Resum

L'objectiu principal d'aquest treball estudiar el problema de tres cossos restringit i el model bicircular, juntament amb l'obtenció de diverses òrbites periòdiques en el darrer model. Del problema de tres cossos restringit se'n tracta la deducció de les equacions de moviment en coordenades sinòdiques, les corbes de velocitat zero, els punts d'equilibri i la seva estabilitat lineal. Per altra banda, del model bicircular només se'n realitza una deducció de les equacions de moviment. Posteriorment, es defineixen diversos algoritmes numèrics per acabar l'escrit amb la cerca d'òrbites periòdiques en el model bicircular.

2020 Mathematics Subject Classification. 70F07, 70F15, 65L06, 65L05

Agraïments

Vull agrair, en primer lloc, al Dr. Àngel Jorba per tot el suport, temps i entusiasme que m'ha transmès al llarg d'aquest darrer any de treball.

També voldria agrair a la meva família i amics, en especial al Guillermo i a la Marta, que m'han acompanyat gran part de la meva vida i, per si no n'hi hagués prou, durant el grau de Matemàtiques.

I, en últim lloc, a l'Ona. Qui és el meu Sol del dia a dia.

Índex

1 Introducció

El projecte

El problema dels N-cossos ha sigut un dels objectes d'estudi que més ha ocupat als físics i matemàtics al llarg dels últims segles. De fet, les primeres grans aportacions s'atribueixen a Newton amb la llei de gravitació universal, que va permetre definir-lo tal com el coneixem avui en dia.

El problema dels N-cossos té multitud d'aplicacions en la mecànica celeste. Aquest ajuda a descriure tant el moviment orbital dels satèl·lits al voltant d'un planeta, com els llançaments de cossos a l'espai (com per exemple, enviar una sonda a Mart). I fins i tot, l'estabilitat o inestabilitat d'una estació espacial situada al voltant de la Terra.

S'ha demostrat que no existeix una solució analítica general pel problema dels N-cossos. De fet, només se'n coneix la solució del problema per $N = 2$ i alguns casos particulars per $N = 3$. Altrament, s'ha de recórrer a diverses estratègiques de càlcul numèric per a obtenir aproximacions de la solució.

No obstant això, existeix molts casos aplicables a la naturalesa de l'espai on la força d'atracció gravitacional del sistema està quasi totalment generada per dues o tres masses. En aquests casos podem negligir les altres masses i treballar com si el sistema, que inicialment pot ser molt complex, ens fos descrit per un problema de N-cossos per a una $N = 2, 3$. D'aquesta manera, podem aplicar la teoria del problema de dos o tres cossos per a realitzar aproximacions prou precises del moviment d'una partícula en el sistema desitjat.

Precisament, l'objectiu d'aquest treball no serà altre que estudiar-ne diversos models que s'utilitzen per a predir el moviment d'una partícula en l'espai. En concret, ens centrarem en el problema de tres cossos restringit planar i circular, un cas particular del problema de tres cossos general. També, estudiarem el model bicircular el qual permet aproximar el moviment d'una partícula, de massa negligible, afectat per la força de gravitació de tres masses. Finalment, acabarem cercant diverses òrbites periòdiques en el model bicircular, emprant la teoria del problema de tres cossos restringit i diversos mètodes numèrics que definirem.

Estructura de la Memòria

Podríem dir que aquest treball està diferenciat en dues parts. La primera part, que engloba el capítol 2 i 3, consta d'un estudi teòric del problema de tres cossos restringit i el model bicircular. La segona part, de caràcter més numèric, engloba el capítol 4 i 5 on definim els diferents mètodes per obtenir les òrbites periòdiques i, posteriorment, comentem els resultats obtinguts. Finalment, tanquem l'escrit amb les conclusions del treball.

A continuació descriurem els continguts de cada capítol:

Capítol 2. En aquest capítol realitzem un estudi bàsic del problema de tres cossos restringit planar i circular. Comencem obtenint les equacions de moviment en coordenades sinòdiques per a posteriorment, estudiar-ne els punts d'equilibri i la seva estabilitat lineal. Finalitzem el capítol, presentant una característica única del problema de tres cossos restringit, les corbes de velocitat zero.

Capítol 3. Aquí presentem el model bicircular tot i que l'objectiu final és deduir les equacions de moviment.

Capítol 4. El propòsit del capítol 4 és recopilar tots els mètodes numèrics i algoritmes que utilitzarem per al programari de cerca d'òrbites periòdiques. Començarem explicant els mètodes Runge-Kutta, en concret, el Runge-Kutta 4-5. Posteriorment, definirem un algoritme de cerca que farem servir com a base per a l'obtenció d'òrbites periòdiques juntament amb un mètode de continuació. Finalitzarem el capítol explicant el mètode de Tir múltiple.

Capítol 5. En aquest apartat trobarem les òrbites periòdiques obtingudes, primer al voltant d'un punt estable i, posteriorment, en un d'inestable.

Capítol 6. Aquest darrer capítol conté les conclusions del treball.

2 Problema de tres cossos restringit

Les primeres aparicions del problema de tres cossos restringit ja s'associen al segle XVII al llibre *Philosophiæ Naturalis Principia Mathematica* d'Isaac Newton, on estudia una possible estabilitat en el sistema Terra-Lluna-Sol. Malgrat això, no va ser fins a mitjans del segle XVIII quan el nom de *problema de tres cossos* es va instaurar com un dels grans camps d'estudi de l'astronomia que dura fins al dia d'avui[2]. L'objectiu principal d'aquest capítol serà presentar un estudi bàsic del problema de tres cossos restringit planar i circular.

Naturalment, la teoria de tres cossos està relacionada amb el problema de dos cossos i la teoria de la força de camp central. Ambdues s'han vist a l'assignatura de Modelització al grau de Matemàtiques i, per tant, suposarem tots els resultats per coneguts. Tanmateix, recordarem un dels resultats més importants: Podem simplificar el problema de dos cossos com a dos problemes de camp central. En particular, el problema de dos cossos en té de solució analítica.

Ara bé, si considerem una partícula més en el nostre sistema, les equacions de moviment es compliquen molt més degut a la força d'atracció que creada per la tercera massa, fins al punt de no poder-les reescriure com teníem en el problema de dos cossos [3]. Fixem-nos, però, que si la massa m_3 fos nul·la implicaria que la força d'atracció associada a la tercera massa seria també nul·la, i, en definitiva, el moviment de m_1 i m_2 romandria igual. Restaria estudiar que passaria amb el moviment de la tercera massa m_3. D'aquesta forma, arribem a la definició següent:

Definició 2.1. *Siguin m_1, m_2 dos cossos amb massa no nul·la i òrbites circulars al voltant del seu centre de masses. Anomenem* ***problema restringit de tres cossos*** *a determinar el moviment d'un tercer cos amb $m_3 \ll m_2 < m_1$. Anomenarem* ***primaris*** *als cossos m_1 i m_2.*

Aquesta és una definició més general del cas que ens dedicarem a estudiar al llarg del capítol. En concret, suposarem que el moviment de les masses primàries és circular respecte del centre de masses i, a més, que el moviment de la tercera massa es troba contingut en el pla de moviment de les altres dues. És a dir, treballarem en les hipòtesis del **problema de tres cossos restringit, planar i circular**.

2.1 Equacions de moviment

En aquesta secció obtindrem les equacions de moviment de la massa m_3 en el sistema inercial clàssic (O, X, Y) i les transformarem mitjançant una rotació que definirem dins d'un sistema de coordenades rotatòries anomenat sistema sinòdic. L'avantatge de treballar amb aquest nou sistema de coordenades no és només la significant simplificació de les posicions de les masses primàries sinó, també, el fet que les equacions de moviment disposin d'una integral primera. Tanmateix, no serà fins a la següent secció on estudiarem la informació que ens proporciona la integral primera. Comencem, ara sí, amb la deducció de les equacions de moviment.

Suposem, doncs, que estem en les hipòtesis del problema de tres cossos restringit planar

[2] Al darrer segle s'han trobat diverses estratègies numèriques per aproximar solucions del problema de tres cossos.

[3] Aquí i al llarg del treball sempre considerarem la força d'atracció amb la seva definició Newtoniana.

i circular. Anomenem les dues partícules primàries per m_1 i m_2 i el seu centre de masses per O. Per l'assignatura de Modelització sabem que les masses primàries, que es mouen circularment respecte al centre de masses O, tenen velocitat constant.

Les equacions de moviment de m_3 en el sistema de coordenades cartesianes (O, X, Y) venen definides per la força F sotmesa a la partícula:

$$\frac{d^2X}{dt^{*2}} = \frac{\partial F}{\partial X}, \qquad \frac{d^2Y}{dt^{*2}} = \frac{\partial F}{\partial Y}.$$

Aquesta força aplicada a la partícula és, íntegrament, la força gravitatòria generada per les masses m_1 i m_2. Considerant la constant de gravitació Gaussiana k podem escriure la força com:

$$F = k^2 \left(\frac{m_1}{R_1} + \frac{m_2}{R_2} \right),$$

on R_1 i R_2 són les distàncies de les masses m_1 i m_2 a la massa m_3 respectivament. Definirem la subseqüent notació per simplificar els pròxims passos.

Definició 2.2. *Anomenarem per* $\boldsymbol{l}$ *a la distància entre les masses* m_1 *i* m_2*,* $\boldsymbol{a}$ *per la distància entre la partícula* m_2 *i el centre de masses* O*.* $\boldsymbol{b}$ *per la distància entre la partícula* m_1 *i* O*. Finalment denotarem per* $\boldsymbol{n}$ *el moviment mitjà de la partícula* m_3*.*

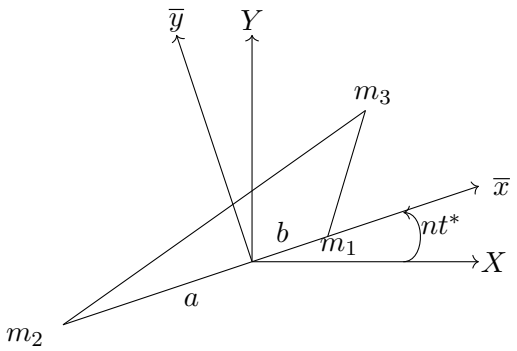


Figura 1: Plantejament del sistema de referència.

Seguint la figura 1 és clar que els mòduls de les distàncies R_1 i R_2 són:

$$R_1 = \left[(X - X_1)^2 + (Y - Y_1)^2 \right]^{1/2},$$
$$R_2 = \left[(X - X_2)^2 + (Y - Y_2)^2 \right]^{1/2},$$

on les coordenades (X_1, Y_1) i (X_2, Y_2) representen la posició de m_1 i m_2 que varia amb el temps respectivament. Les masses primàries tenen òrbita circular amb la velocitat angular constant, per tant;

$$X_1 = b\cos nt^*, \quad Y_1 = b\sin nt^*,$$
$$X_2 = -a\cos nt^*, \quad Y_2 = -a\sin nt^*.$$

Proposició 2.3. *Les equacions del moviment de m_3 són:*

$$\frac{d^2X}{dt^{*2}} = -k^2\left[\frac{m_1(X - b\cos nt^*)}{R_1^3} + \frac{m_2(X + a\cos nt^*)}{R_2^3}\right],$$
$$\frac{d^2Y}{dt^{*2}} = -k^2\left[\frac{m_1(Y - b\sin nt^*)}{R_1^3} + \frac{m_2(Y + a\sin nt^*)}{R_2^3}\right]. \tag{2.1}$$

Demostració:

Expressem R_1 i R_2 amb les coordenades (X_1, Y_1), (X_2, Y_2):

$$R_1 = \left[(X - b\cos nt^*)^2 + (Y - b\sin nt^*)^2\right]^{1/2},$$

$$R_2 = \left[(X + a\cos nt^*)^2 + (Y + a\sin nt^*)^2\right]^{1/2}.$$

L'expressió de la força F queda com,

$$F = k^2\left(\frac{m_1}{[(X - b\cos nt^*)^2 + (Y - b\sin nt^*)^2]^{1/2}} + \frac{m_2}{[(X + a\cos nt^*)^2 + (Y + a\sin nt^*)^2]^{1/2}}\right).$$

Calculem les derivades parcials de F i obtenim el resultat. □

L'expressió explícita del temps a les equacions del moviment que hem obtingut no ens permetrà obtenir una integral primera. Una estratègia molt útil d'abordar el problema en aquests casos és canviar el sistema de referència per un més adient. I bé, ¿que podríem dir que és *adient*? Com que l'expressió explícita del temps ve causada pel moviment de les masses primàries, sembla natural pensar que un sistema de referència que deixi fix m_1 i m_2 ens donarà una expressió millor de les equacions de moviment de m_3.

Definició 2.4. *El **sistema sinòdic** és un sistema de referència rotacional, el qual té com a origen el baricentre de les masses i per a rotació, la que deixa fixes les masses primàries.*

A la figura 1, (O, X, Y) és el clàssic sistema de referència fix amb origen al centre de masses O. Per altra part, $\overline{x}$ i $\overline{y}$ representen els eixos del sistema sinòdic, que dintre del sistema de referència (O, X, Y) roten amb un angle de nt^*, essent l'angle recorregut per m_1 respecte al eix de les x's.

Proposició 2.5. *La transformació del sistema sinòdic és la següent:*

$$X = \overline{x}\cos nt^* - \overline{y}\sin nt^*,$$
$$Y = \overline{x}\sin nt^* + \overline{y}\cos nt^*.$$

On, amb notació matricial $R = A\overline{r}$

$$A = \begin{pmatrix}\cos nt^* & -\sin nt^* \\ \sin nt^* & \cos nt^*\end{pmatrix}.$$

Demostració

El moviment de les masses m_1 i m_2 és circular amb la velocitat angular constant. Com hem comentat anteriorment, la figura 1 mostra que l'angle de rotació en un temps determinat t^* és nt^* i, per tant, la velocitat angular és nt^*. En definitiva, la matriu que ens produeix el canvi de referència és la matriu de rotació A. □

Proposició 2.6. *Les equacions de moviment (2.1) en coordenades sinòdiques són:*

$$\frac{d^2\overline{x}}{dt^{*2}} - 2n\frac{d\overline{y}}{dt^*} - n^2\overline{x} = -k^2\left[m_1\frac{(\overline{x}-b)}{\overline{r}_1^3} + m_2\frac{(\overline{x}-a)}{\overline{r}_2^3}\right],$$
$$\frac{d^2\overline{y}}{dt^{*2}} + 2n\frac{d\overline{x}}{dt^*} - n^2\overline{y} = -k^2\left[\frac{m_1\overline{y}}{\overline{r}_1^3} + \frac{m_2\overline{y}}{\overline{r}_2^3}\right]. \tag{2.2}$$

On $\overline{r}_1$ i $\overline{r}_2$ representa respectivament R_1 i R_2 sense dependència explícita en el temps.

Demostració.

Per demostrar-ho, identificarem els eixos amb el pla complex. Sigui $Z = ze^{int}$ on:

$$z = \overline{x} + i\overline{y}, \quad Z = X + iY, \quad i = \sqrt{-1}, \quad \left|e^{int}\right| = 1.$$

Essent $Z_1 = be^{int}$ i $Z_2 = -ae^{int}$ tenim que,

$$R_1 = |Z - Z_1| = |z - b| = \left[(\overline{x}-b)^2 + \overline{y}^2\right]^{1/2}.$$

Anàlogament, per R_2:

$$R_2 = |Z - Z_2| = |z + a| = \left[(\overline{x}+a)^2 + \overline{y}^2\right]^{1/2}.$$

Recordem que estem buscant una equació del moviment del tipus $\frac{d^2Z}{dt^{*2}} = \frac{dF}{dZ}$ on $F = k^2\left(\frac{m_1}{R_1} + \frac{m_2}{R_2}\right)$.
Fent ús de la regla de la cadena:

$$\frac{d^2Z}{dt^{*2}} = \left(\frac{dz}{dt^*}e^{int^*} + z(in)e^{int^*}\right)' = \left(\frac{dz^2}{dt^{*2}} + 2in\frac{dz}{dt^*} - n^2z\right)e^{int^*}.$$

Per altra part, si apliquem les expressions de R_1 i R_2 a F obtenim:

$$F = k^2\left(\frac{m_1}{R_1} + \frac{m_2}{R_2}\right) = k^2\left(\frac{m_1}{|z-b|} + \frac{m_2}{|z+a|}\right) = -k^2\left[m_1\frac{(z-b)}{|z-b|^3} + m_2\frac{(z+a)}{|z+a|^3}\right].$$

D'aquesta manera l'equació ens queda per:

$$\left(\frac{dz^2}{dt^{*2}} + 2in\frac{dz}{dt^*} - n^2z\right) = -k^2\left[m_1\frac{(z-b)}{|z-b|^3} + m_2\frac{(z+a)}{|z+a|^3}\right].$$

Agafant part real i imaginària obtenim les equacions desitjades. □

Aquestes equacions ja ens permetran calcular una integral primera. Com a conseqüència obtindrem una nova constant que serà molt útil al llarg de les pròximes seccions.

2.2 La constant de Jacobi

En aquesta secció definirem la constant de Jacobi i comentarem la seva principal conseqüència. Abans, però, establirem unitats de massa, temps i distància de forma astuta per simplificar tant les unitats de les variables com les mateixes equacions resultants.

A partir d'ara definirem la massa total del sistema com la unitat de massa i la distància entre les masses m_1 i m_2 com la nostra unitat de distància. Així doncs, obtindrem una

relació entre les masses i distàncies del nostre sistema. També extraure'm la dependència del temps de les equacions posant com a unitat de temps el moment mitjà. En definitiva, amb les noves consideracions tenim que:

$$M = 1 \to m_1 = 1 - \mu, \quad m_2 = \mu,$$

$$l = 1, \quad t = nt^*.$$

Observació 2.7. Utilitzant el sistema de referència sinòdic i les unitats prèviament definides, les coordenades de les masses m_1 i m_2 se simplifiquen molt. De fet, l'esquema resultant del problema de tres cossos seguirà el de la figura 2.

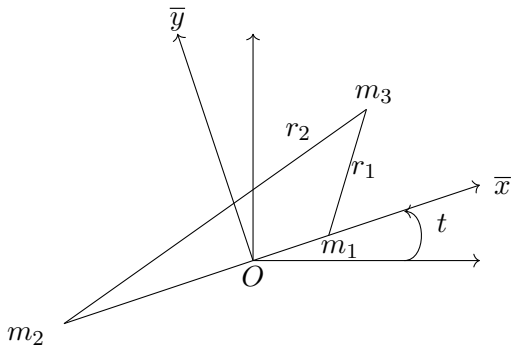


Figura 2: Esquema de les posicions de les masses amb les noves unitats.

Efectivament, si situem el centre de masses O en el centre de coordenades $(0, 0)$ la massa $m_1 = 1-\mu$ és trobarà a distància μ del origen i, per tant, tindrà com a coordenades $(\mu, 0)$. Per altra part, la massa $m_2 = \mu$ és trobarà a distància $1 - \mu$ del origen amb coordenades $(\mu - 1, 0)$. Cal comentar que l'elecció de posar a la dreta o a l'esquerra una massa o bé l'altre és totalment irrellevant per la naturalesa del nostre problema. Al llarg del treball es considerarà únicament les posicions abans descrites i no pas d'altres configuracions.

Proposició 2.8. *Amb les noves unitats les equacions de moviment (2.2) queden com:*

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

$$\ddot{y} + 2\dot{x} = y\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right). \tag{2.3}$$

Considerant una certa funció Ω, podem escriure les equacions de forma compacta:

$$\ddot{x} - 2\dot{y} = \Omega_x,$$

$$\ddot{y} + 2\dot{x} = \Omega_y. \tag{2.4}$$

Demostració.

Partim doncs de les equacions (2.2):

$$\frac{d^2\overline{x}}{dt^{*2}} - 2n\frac{d\overline{y}}{dt^*} - n^2\overline{x} = -k^2\left[m_1\frac{(\overline{x}-b)}{\overline{r}_1^3} + m_2\frac{(\overline{x}-a)}{\overline{r}_2^3}\right],$$

$$\frac{d^2\overline{y}}{dt^{*2}} + 2n\frac{d\overline{x}}{dt^*} - n^2\overline{y} = -k^2\left[\frac{m_1\overline{y}}{\overline{r}_1^3} + \frac{m_2\overline{y}}{\overline{r}_2^3}\right].$$

La igualtat esquerra d'ambdues equacions romandrà sense canvis tret que simplificarem el valor n per 1, ja que $t = nt^*$. Per la part dreta de les equacions sabem per l'observació d'abans que $m_1 = 1 - \mu$, $m_2 = \mu$, $a = \mu - 1$ i $b = \mu$. Per tant, l'única part no trivial serà veure que $k = 1$.

Com que $m_3 = 0$, el balanç entre força centrífuga i gravitatòria ens dona,

$$k^2 \frac{m_1 m_2}{l^2} = m_2 a n^2 = m_1 b n^2.$$

D'aquí obtenim,

$$k^2 m_1 = a n^2 l^2, \quad k^2 m_2 = b n^2 l^2.$$

Sumant, com que $a + b = l$

$$k^2(m_1 + m_2) = n^2(a + b)l^2 = n^2 l^3.$$

Finalment, com que hem definit $l = 1$, $m_1 + m_2 = 1$ i el moment mitjà $n = 1$,

$$k^2 = n^2 l^3 = 1.$$

Ara si, podem simplificar les equacions

$$\begin{cases} \ddot{x} - 2\dot{y} - x = -\left[(1-\mu)\frac{(x-\mu)}{r_1^3} + \mu\frac{(x-\mu+1)}{r_2^3}\right], \\ \ddot{y} + 2\dot{x} - y = -\left[\frac{(1-\mu)y}{r_1^3} + \frac{\mu y}{r_2^3}\right]. \end{cases} \tag{2.5}$$

On,

$$r_1^2 = (x-\mu)^2 + y^2,$$
$$r_2^2 = (x+1-\mu)^2 + y^2.$$

Finalment, hem arribat a les equacions que buscàvem. Per escriure-ho de forma compacte ens caldrà trobar la funció Ω tal que les seves derivades parcials coincideixin amb la part dreta de les equacions. En definitiva, la funció que busquem és:

$$\Omega := \frac{1}{2}\left(x^2 + y^2\right) + \frac{\mu}{r_2} + \frac{1-\mu}{r_1} + \frac{1}{2}\mu\left(1-\mu\right). \tag{2.6}$$

□

Les equacions (2.4) admeten una integral primera i, com a conseqüència, una nova constant. Per trobar-la hem de multiplicar la primera equació per $\dot{x}$ i la segona per $\dot{y}$:

$$\dot{x}\ddot{x} - 2\dot{y}\dot{x} = \Omega_x \dot{x},$$

$$\dot{y}\ddot{y} + 2\dot{x}\dot{y} = \Omega_y \dot{y}.$$

D'on sumant obtindrem $\dot{y}\ddot{y} + \dot{x}\ddot{x} = \frac{d\Omega}{dt}$. I si integrem respecte al temps, ens resultarà que:

$$\dot{x}^2 + \dot{y}^2 = 2\Omega - C.$$

Definició 2.9. *Definim* ***la integral de Jacobi*** *com l'expressió* $2\Omega - \dot{x}^2 - \dot{y}^2$ *i la* ***constant de Jacobi*** *al valor* C *tal que* $C = 2\Omega - \dot{x}^2 - \dot{y}^2$.

Observació 2.10. L'expressió anterior ens permet relacionar la velocitat amb la Constant de Jacobi. Ja que $v^2 = \dot{x}^2 + \dot{y}^2$,

$$v^2 = 2\Omega - C.$$

Al contrari que en el cas del problema de camp central o bé el problema de dos cossos, en el problema de tres cossos restringit no es compleix la llei de conservació d'energia. Això és degut a la simplificació que fem al suposar que la massa de la tercera partícula sigui igual a zero. Per sort, la constant de Jacobi ens substituirà aquesta mancança i ens permetrà trobar molta més informació sobre el moviment de la massa m_3. Tanmateix, no serà fins a la secció **2.5** que estudiarem la principal conseqüència d'aquesta constant.

2.3 Punts d'equilibri

En el problema de tres cossos restringit els punts on la partícula m_3 romandrà en repòs es troben quan aquesta experimenta un balanç entre les forces (considerant el sistema de coordenades sinòdic). Al llarg d'aquesta secció demostrarem l'existència de cinc punts d'equilibri en el problema de tres cossos restringit per a una μ qualsevol.

Suposem, doncs, que (x, y) és un punt d'equilibri. Llavors és clar que les derivades $\dot{x}$, $\dot{y}$ i les derivades segones $\ddot{x}$ i $\ddot{y}$ són igual a zero. Utilitzant l'equació diferencial (2.4) veiem que,

$$\begin{aligned} \ddot{x} - 2\dot{y} &= \Omega_x = 0, \\ \ddot{y} + 2\dot{x} &= \Omega_y = 0. \end{aligned}$$

Desfent la notació compacta, tenim:

$$\begin{cases} x - \frac{(1-\mu)(x-\mu)}{r_1^3} - \frac{\mu(x+1-\mu)}{r_2^3} = 0, \\ y\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right) = 0. \end{cases} \tag{2.7}$$

Ara l'existència dels punts d'equilibri en el problema de tres cossos queda reescrita com l'existència de solucions del sistema (2.7). Notem per la segona equació que tenim dos casos segons si el valor de y és igual o diferent de zero.

2.3.1 Cas $y = 0$

Si suposem que $y = 0$ només haurem de resoldre la primera equació:

$$x - \frac{(1-\mu)(x-\mu)}{r_1^3} - \frac{\mu(x+1-\mu)}{r_2^3} = 0,$$

on els valors de r_1 i r_2 són:

$$\begin{aligned} r_1 &= (x - \mu), \\ r_2 &= (x + 1 - \mu). \end{aligned}$$

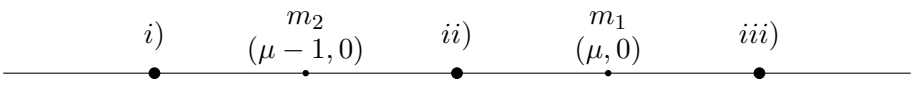
Per tant, haurem de resoldre:

$$x - \frac{(1-\mu)}{(x-\mu)^2} - \frac{\mu}{(x+1-\mu)^2} = 0. \tag{2.8}$$

Per a cada problema de tres cossos associat a la massa μ obtindrem una equació de cinquè grau associada a l'equació anterior. Fixem-nos que al suposar que el valor de la coordenada $y = 0$ estem considerant que totes les possibles solucions de (2.8) es troben contingudes en la recta que passa per les masses primàries. Podem trobar-nos doncs en les següents possibilitats:

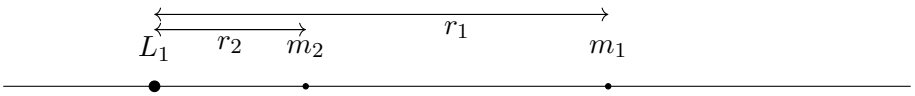
i) El punt es troba abans de la primera massa $x_* \in (-\infty, \mu - 1)$.
ii) El punt es troba entre les dues masses $x_* \in (\mu - 1, \mu)$.
iii) El punt es troba després de les dues masses $x_* \in (\mu, \infty)$.

i) Primer interval, el punt es troba a l'esquerra de les dues masses.



Proposició 2.11. *En l'interval $x_* \in (-\infty, \mu - 1)$ existeix un únic punt d'equilibri que anomenarem L_1.*

Demostració.



Definint r_1 i r_2 com les distàncies del punt L_1 fins a les masses m_1 i m_2 respectivament i, denotant $r_2 := e$, podem reescriure la nostra equació en termes d'èpsilon. En efecte,

$$r_2 = \epsilon \to r_1 = 1 + \epsilon, \quad x_* = \mu - 1 - \epsilon.$$

Substituint a (2.8):

$$\epsilon + 1 - \mu + \frac{\mu - 1}{(1+\epsilon)^2} - \frac{\mu}{\epsilon^2} = 0.$$

$$\epsilon^3 + \epsilon^2 - \mu\epsilon^2 + \frac{(\mu - 1)\epsilon^2}{(1+\epsilon)^2} - \mu = 0.$$

$$\epsilon^3(1+\epsilon)^2 + \epsilon^2(1+\epsilon)^2 - \mu\epsilon^2(1+\epsilon)^2 + (\mu - 1)\epsilon^2 - \mu(1-\epsilon)^2 = 0.$$

D'on finalment, obtenim l'equació de cinquè grau:

$$\epsilon^5 + (3 - \mu)\epsilon^4 + (3 - 2\mu)\epsilon^3 - \mu\epsilon^2 - 2\mu\epsilon - \mu = 0. \tag{2.9}$$

Sabem per la teoria de Galois que les equacions algebraiques de cinquè grau no tenen una expressió específica per obtenir les arrels. No obstant això, existeixen diverses estratègies numèriques per a calcular arrels de polinomis de grau 5 o superior, de fet, en els annexos **A** utilitzarem el mètode de Newton per a calcular L_1 segons una $\mu = \mu_0$ donada. De totes maneres, per a demostrar l'existència d'una arrel real positiva a l'equació (2.9) per un rang de $\mu \in (0, \frac{1}{2})$ ens caldrà utilitzar **regla dels signes de Descartes**.

Proposició 2.12. ***Regla dels signes Descartes****. Sigui $p(x)$ un polinomi amb coeficients en $\mathbb{R}$ sigui z_p el nombre d'arrels reals positives del polinomi i c el nombre de canvis de signe en els termes (ordenats de major grau a menor). Llavors es compleix que $z_p \leq c$ i la diferència és un nombre parell no negatiu.*

Demostració.

Hem de demostrar que $\forall p(x)$ polinomi amb coeficients reals $\exists k \in \mathbb{N} \cup \{0\}$ tal que $c - z_p = 2k$.

Suposem sense perdre generalitat que $p(x)$ és mònic, ja que si no ho fos, podríem dividir el polinomi pel coeficient que acompanya la x de grau màxim. Podem suposar també que $p(x)$ tindrà un terme independent a_0 , ja que, novament, si tinguéssim $a_0 x^b$ per alguna $b \in \mathbb{N}$, podríem dividir el polinomi pel terme x^b. Notem que aquest procés no alterarà el nombre d'arrels positives. En definitiva, el nostre polinomi tindrà la forma:

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0, \quad a_i \in \mathbb{R}, a_0 \neq 0, \quad \forall i \in \{0, \ldots, n-1\}. \quad (2.10)$$

Veiem primer que el nombre de canvis c és parell o imparell depenen del signe del terme independent. En efecte, suposem que a_0 és positiu, llavors al nombre de canvis de signe serà parell i si a_0 és negatiu, c serà imparell. Això és degut al fet que estem considerant només els signes i, per tant, tenim dos estats possibles $(+, -)$. Com que el polinomi $p(x)$ és mònic, estem començant per a l'estat $+$, llavors si acabem pel mateix estat haurem realitzat $2n$ canvis on $n \in \mathbb{N} \cup \{0\}$ o bé si l'estat ha canviat llavor tenim $2n + 1$ canvis, novament per $n \in \mathbb{N} \cup \{0\}$.

També podem veure la paritat del nombre d'arrels reals positives respecte al signe del terme independent. En concret, passarem a demostrar que quan el terme independent és positiu el polinomi té un nombre parell d'arrels reals positives i quan el terme independent és negatiu, en té un nombre imparell. Per tal de fer-ho farem servir inducció respecte al grau del polinomi n.

Cas base $n = 1$. Sigui $p(x) = x \pm a$ amb $a \in \mathbb{R} \setminus \{0\}$. Considerem primer el cas $p(x) = x + a$. Fixem-nos que en el rang $x \in [0, +\infty)$ el polinomi només pren valors positius i, per tant, no tenim cap arrel real positiva. Pel cas $p(x) = x - a$ clarament tallem un cop l'eix de les x's, essent doncs imparell el nombre d'arrels reals positives.

Suposem que la propietat és certa per a polinomis de grau menor que n. Demostrarem que també ho és pels polinomis de grau n. Començarem pel cas $a_0 < 0$. Com que $\lim_{x\to\infty} p(x) \to +\infty$ i $p(0) = a_0 < 0$ pel teorema de Bolzano és clar que el polinomi té alguna arrel real positiva r. Per tant, el podem reescriure com $p(x) = (x - r)q(x)$ on $q(x)$ el polinomi resultant de dividir $p(x)$ pel factor irreductible $(x - r)$ (recordem que estem treballant amb totes les operacions en el cos dels reals). El polinomi $q(x)$ té grau $n - 1$, és mònic i té per terme independent $b_0 > 0$ perquè $a_0 = -rb_0 < 0$, de manera que per la hipòtesi d'inducció, $q(x)$ té un nombre parell d'arrels reals positives. En definitiva, $p(x)$ té un nombre imparell d'arrels reals positives.

El cas on $a_0 > 0$ és molt semblant. Com que $p(0) > 0$ no podem aplicar el teorema de Bolzano directament, però podem distingir dos casos. El primer cas és que polinomi no tingui arrels reals positives, per tant, ja hauríem acabat. El segon cas, és que en tingui com a molt una i, en conseqüència, podem aplicar l'argument anterior. En efecte, si $p(x)$ té una arrel real positiva r, llavors podem reescriure $p(x) = (x - r)q(x)$, on novament $q(x)$ és el polinomi resultant de la divisió $p(x)$ pel factor irreductible $(x - r)$. D'aquesta manera, podem aplicar la hipòtesi d'inducció i, com que el terme independent b_0 de $q(x)$

és negatiu, tindrem aplicant la hipòtesi d'inducció que $q(x)$ té un nombre imparell d'arrels reals positives. En conclusió, $p(x)$ tindrà un nombre parell d'arrels reals positives.

Així doncs, és clar que c i z_p tenen la mateixa paritat. Resta veure que $c \geq z_p$. Novament, utilitzarem inducció sobre n, el grau del polinomi. El cas base $n = 1$ és clar que és cert. Suposem doncs la validesa per a polinomis de grau $n-1$ i mirem que per un polinomi de grau n es compleix que $c \geq z_p$. Si ens fixem en el polinomi resultant de derivar $p(x)$, tenim que el nombre de canvis de signe c' és c o bé $c-1$ i que el nombre d'arrels positives z_p' és $z_p - 1$ com a conseqüència del teorema de Rolle. Podem aplicar la hipòtesi d'inducció al polinomi derivat, tot plegat tenim,

$$z_p - 1 \leq z_p' \leq c' \leq c.$$

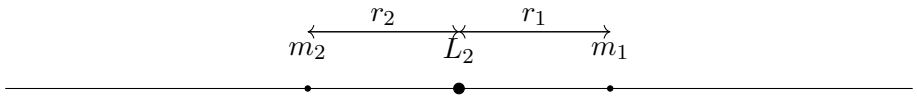
Llavors $c - z_p \geq -1$ i que $c - z_p = 2k$, en definitiva, obtenim que la diferència ha de ser positiva. □

Finalment, per la regla de signes de Descartes l'equació (2.9), que conté un canvi de signe, té una única solució real positiva (com a màxim hi ha una i com la diferència entre $c - z_p = 2k$ per un k enter, és clar que ha de ser que $k = 0 \to c = z_p$). Per tant, al ser ϵ una distància, l'arrel trobada serà l'única solució vàlida, essent L_1. □

ii) Segon interval, el punt es troba entre les dues masses.

Proposició 2.13. *En l'interval $x_* \in (\mu - 1, \mu)$ existeix un únic punt d'equilibri que anomenarem L_2.*

Demostració.



Definint ϵ tal que $r_2 = \epsilon$, $r_1 = 1 - \epsilon$ i $x = \mu - 1 + \epsilon$, l'equació (2.8) queda com:

$$\epsilon - 1 + \mu + \frac{1-\mu}{(\epsilon-1)^2} - \frac{\mu}{\epsilon^2} = 0.$$

Com en el cas anterior haurem de multiplicar pels dos denominadors, obtenint:

$$\epsilon^5 - (3-\mu)\epsilon^4 + (3-2\mu)\epsilon^3 - \mu\epsilon^2 + 2\mu\epsilon - \mu = 0. \tag{2.11}$$

En aquest cas, la Regla de Descartes i el canvi de signe que comporta (2.11) ens assegura l'existència de com a mínim una arrel real positiva.

Per a demostrar la unicitat seguirem la idea descrita al llibre de Pollard [7]. Així doncs, reescriure les equacions $\Omega_x = 0$ i $\Omega_y = 0$ en coordenades bipolars $p_1 = r_1$ i $p_2 = r_2$ del punt (x, y). Donat que i $p_1^2 = (x-\mu)^2 + y^2$ i $p_2^2 = (x+1-\mu)^2 + y^2$, podem reescriure $\Omega(x, y)$ per:

$$\Omega(p_1, p_2) = (1-\mu)(\frac{1}{2}p_1^2 + p_1^{-1}) + \mu(\frac{1}{2}p_2^2 + p_2^{-1})$$

Llavors, les equacions $\Omega_x = 0$ i $\Omega_y = 0$ queden com:

$$(1-\mu)\left[p_1 - \frac{1}{p_1^2}\right]\frac{x+\mu}{p_1} + \mu\left[p_2 - \frac{1}{p_2^2}\right]\frac{x-1+\mu}{p_2} = 0$$

$$(1-\mu)\left[p_1-\frac{1}{p_1^2}\right]\frac{y}{p_1}+\mu\left[p_2-\frac{1}{p_2^2}\right]\frac{y}{p_2}=0 \tag{2.12}$$

Naturalment, com que ens trobem en el cas $y=0$ només ens caldrà la primera equació de (2.12). En particular, podem utilitzar que $p_1=x+\mu=p$ i $p_2=1-x-\mu=1-p$, per tant:

$$(1-\mu)\left[p-\frac{1}{p^2}\right]=\mu\left[1-p-\frac{1}{(1-p)^2}\right]$$

Que implica,

$$\frac{1-\mu}{\mu}=\frac{\left[1-p-\frac{1}{(1-p)^2}\right]}{\left[p-\frac{1}{p^2}\right]}$$

Podem definir la funció $F(p)$ tal que

$$F(p):=\frac{\left[1-p-\frac{1}{(1-p)^2}\right]}{\left[p-\frac{1}{p^2}\right]}=\frac{1-\mu}{\mu}\geqq 1\quad \mu\in\left(0,\frac{1}{2}\right]$$

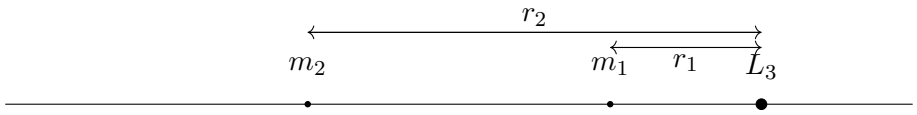
Aquesta funció és estrictament creixent per a l'interval $p\in(0,1)$. A més a més, tenim que $F(\frac{1}{2})=1$ i que $\lim_{p\to 1^-}F(p)\to+\infty$. En definitiva, només hi ha un valor p tal que $F(p)=\frac{1-\mu}{\mu}$ i, per tant, existeix un únic punt d'equilibri. □

Observació 2.14. Fixem-nos que per $\mu\in\left(0,\frac{1}{2}\right)$ aquest punt L_2 sempre es trobarà més a prop de la partícula amb massa més petita, en el nostre cas, m_2. El cas $\mu=\frac{1}{2}$ el punt L_2, com és d'esperar, equidista de les dues masses primàries.

iii) Tercer interval, el punt es troba a la dreta de les dues masses.

Proposició 2.15. *En l'interval $x_*\in(\mu,\infty)$ existeix un únic punt d'equilibri que anomenarem L_3.*

Demostració.



Si definim ϵ tal que $r_1=\epsilon$, $r_2=1+\epsilon$ i $x=\mu+\epsilon$ i l'equació (2.8) queda com:

$$\epsilon+\mu+\frac{1-\mu}{\epsilon^2}-\frac{\mu}{(1+\epsilon)^2}=0.$$

Simplificant els denominadors obtenim:

$$\epsilon^5+(2+\mu)\epsilon^4+(1+2\mu)\epsilon^3-(1-\mu)\epsilon^2-2(1-\mu)\epsilon-(1-\mu)=0.$$

Anàlogament, al cas de L_1, utilitzant el teorema de Descartes veiem que només pot existir una arrel real positiva i, en conseqüència, el punt L_3 □.

En definitiva, pel cas $y=0$ hem trobat tres punts d'equilibri $L_1\in(-\infty,\mu-1)$, $L_2\in(\mu-1,\mu)$ i $L_3\in(\mu,\infty)$ i en sabem que no n'hi ha més. Per altra banda, els valors concrets que determinen la posició de cada punt dependrà únicament del valor μ que considerem.

2.3.2 Cas $y \neq 0$

Tot i com un podria esperar-se el contrari, aquest cas serà molt més simple que l'anterior. Sigui el sistema d'equacions (2.7),

$$\begin{cases} x - \frac{(1-\mu)(x-\mu)}{r_1^3} - \frac{\mu(x+1-\mu)}{r_2^3} = 0, \\ y\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right) = 0. \end{cases}$$

Considerarem $y \neq 0$.

Fixem-nos que les equacions per $r_1 = r_2 = 1$ donen,

$$x - (1-\mu)(x-\mu) - \mu(x+1-\mu) = 0.$$

$$y\,(1 - (1-\mu) - \mu) = 0.$$

Per tant, trobem una solució del sistema. De fet, el nom d'equilateral prové del fet que els dos punts es troben a distàncies $r_1 = r_2 = 1$ de les masses primàries, formant visualment dos triangles equilàters.

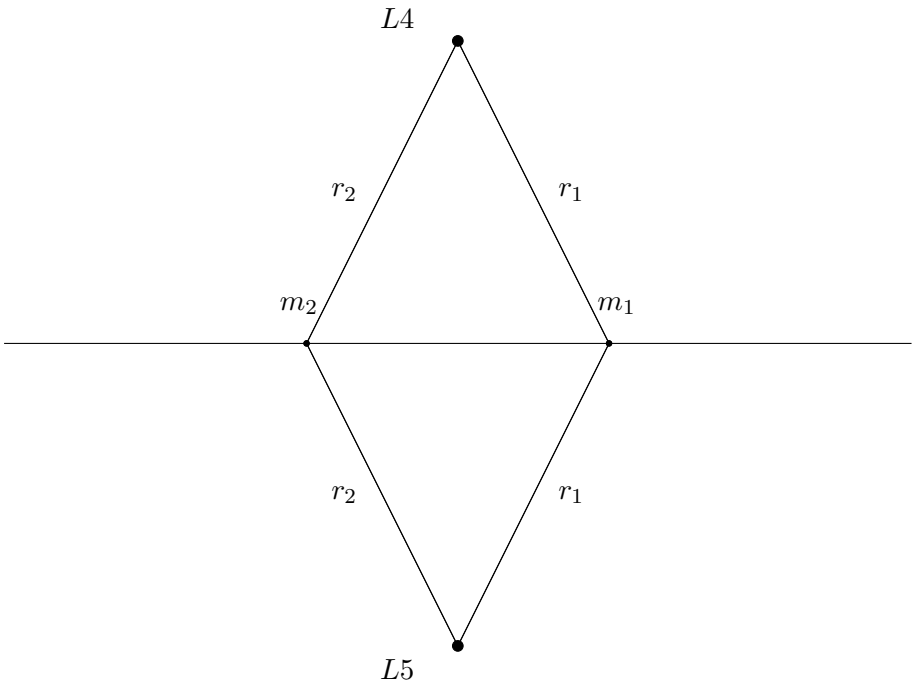


Figura 3: Esquema dels punts L_4 i L_5.

Definició 2.16. *Anomenarem* ***punts de Lagrange*** *als punts d'equilibri del problema de tres cossos restringit. En particular, anomenarem* ***punts de col·lineals*** *a L_1, L_2 i L_3 i* ***punts equilaterals*** *als punts L_4 i L_5.*

El nom de punts de Lagrange s'atribueix al matemàtic i astrònom Joseph-Louis Lagrange que l'any 1772 intentant solucionar el problema de tres cossos va trobar aquestes solucions d'equilibri. No obstant això, els punts de Lagrange varen passar força desapercebuts en l'astronomia fins al segle XX on es varen trobar centenars d'asteroides, els Troians, oscil·lant al voltant del punt L_4 en el sistema Sol-Júpiter i també un nombre

similar d'asteroides, els Grecs, al voltant de la posició L_5. En la pròxima secció veurem precisament que l'estabilitat dels punts equilaterals donen peu a aquest fenomen. Abans, però, veurem un exemple dels punts de Lagrange en el sistema Terra-Lluna que ens serà útil pels pròxims capítols.

2.3.3 Exemple: Punts de Lagrange del sistema Terra-Lluna

En l'apartat **A** dels annexos es pot trobar un càlcul numèric de les coordenades dels punts de Lagrange en el sistema Terra-Lluna. En concret, s'obté

$$m_1 = (0.012505819187, 0), \quad m_2 = (-0.987494180813, 0).$$

$$L1 = (-1.157032814896987, 0), \quad L2 = (-0.83518121199218, 0), \quad L3 = (1.005210650911847, 0).$$

$$L4 = (-0.487494180813, \frac{\sqrt{3}}{2}), \quad L5 = (-0.487494180813, -\frac{\sqrt{3}}{2}).$$

De tal manera que, amb les nostres magnituds obtenim el següent dibuix aproximat:

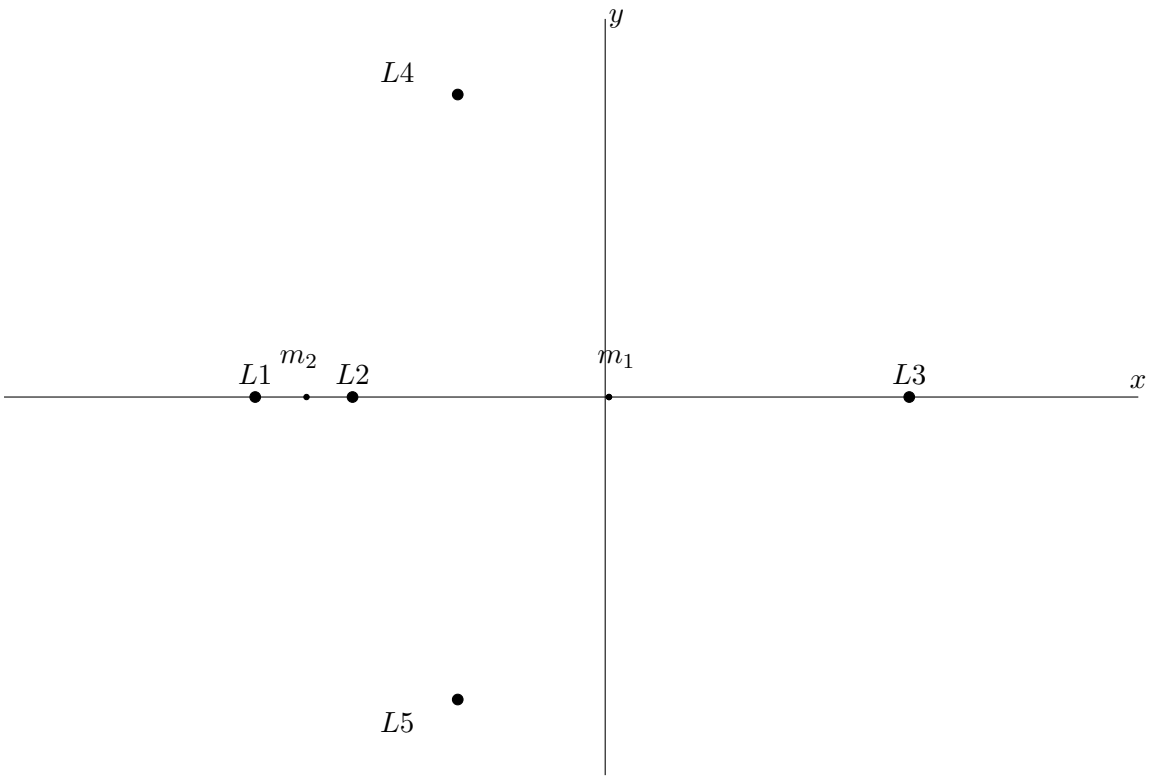


Figura 4: Esquema dels punts de Lagrange pel sistema Terra-Lluna.

2.4 Estabilitat dels punts de Lagrange

Un cop trobada l'existència de cinc punts d'equilibri en el problema de tres cossos restringit, sembla obvi preguntar-se per la seva estabilitat. Aquesta ens donarà informació de la dificultat de trobar-ne d'òrbites periòdiques a prop dels punts de Lagrange, però, igualment, això serà matèria per més endavant. Així doncs, en aquesta secció ens centrarem en l'estudi de l'estabilitat lineal dels punts d'equilibri.

Per a estudiar l'estabilitat és comú extreure polinomi característic de la jacobiana associada al sistema i avaluar els seus valor propis. No obstant això, obtenir la jacobiana amb les equacions de moviment que disposem pot ser un treball certament no trivial. Plantejarem una altra estratègia per aconseguir el polinomi característic que obviï el càlcul de la diferencial. Abans, però, recordem les nocions de punt estable, inestable i asimptòticament estable.

Definició 2.17. *Sigui un sistema d'equacions diferencials,*

$$\dot{x} = X(x).$$

amb un punt d'equilibri $x = a$. La solució $x(t)$ diem que és una ***solució d'equilibri*** *si el camí sobre l'espai x està representat pel punt $x(t_0) = a$.*

Definició 2.18. *La solució $x = a$ és* ***estable*** *si donat un $\epsilon > 0$ existeix un $\delta(\epsilon) > 0$ tal que, si*

$$|x(t_0) - a| \leq \delta,$$

llavors, per tot $t > t_0$

$$|x(t) - a| < \epsilon.$$

Altrament, diem que és ***inestable****. Quan la solució per $t \to \infty$ tendeix al punt d'equilibri diem que és* ***asimptòticament estable****.*

2.4.1 Equacions variacionals de les equacions de moviment

L'objectiu principal d'aquest apartat serà obtenir un anàleg al polinomi característic de la jacobiana del sistema d'equacions de moviment del problema restringit de tres cossos. Recordem que les equacions de moviment (2.4) són:

$$\ddot{x} - 2\dot{y} = \Omega_x,$$

$$\ddot{y} + 2\dot{x} = \Omega_y.$$

Ho reescriurem com a una equació diferencial de primer ordre utilitzant quatre variables tals com:

$$x_1 = x, \quad x_2 = y,$$

$$x_3 = \dot{x}, \quad x_4 = \dot{y}.$$

De tal manera que el sistema queda escrit de la següent forma:

$$\dot{x_1} = x_3, \quad \dot{x_2} = x_4,$$

$$\dot{x_3} = 2x_4 + \frac{\partial \Omega(x_1, x_2)}{\partial x_1}, \quad \dot{x_4} = -2x_3 + \frac{\partial \Omega(x_1, x_2)}{\partial x_2}. \tag{2.13}$$

Les dues primeres equacions fan referència a les components de la velocitat i les altres dues a les de l'acceleració. Per tant, les solucions d'equilibri s'obtenen d'igualar totes les equacions a 0.

$$x_3 = 0, \quad x_4 = 0,$$

$$2x_4 + \frac{\partial \Omega(x_1, x_2)}{\partial x_1} = 0, \quad -2x_3 + \frac{\partial \Omega(x_1, x_2)}{\partial x_2} = 0. \tag{2.14}$$

Òbviament, les solucions són els cinc punts d'equilibri trobats a la secció anterior.

Proposició 2.19. *Siguin $L_i = (a_i, b_i)$ per $i \in \{1,2,3,4,5\}$ els punts de Lagrange. I siguin les variacions $\varepsilon > 0$ i $\eta > 0$ tals que els punts $p_i := (x_i, y_i)$ per $i \in \{1,2,3,4,5\}$ essent*

$$x_i = a_i + \varepsilon, \quad y = b_i + \eta,$$

són punts que pertanyen a un entorn petit dels respectius L_i. Llavors tenim que les equacions variacionals associades a aquestes variacions ε, η són:

$$\begin{aligned} \ddot{\epsilon} - 2\dot{\eta} &= \Omega_{xx}(a,b)\epsilon + \Omega_{xy}(a,b)\eta + O(2), \\ \ddot{\eta} + 2\dot{\epsilon} &= \Omega_{xy}(a,b)\epsilon + \Omega_{yy}(a,b)\eta + O(2). \end{aligned} \tag{2.15}$$

Que tenen com a polinomi característic:

$$\lambda^4 + (4 - \Omega_{xx} - \Omega_{yy})\lambda^2 + \Omega_{xx}\Omega_{yy} - \Omega_{xy}^2 = 0. \tag{2.16}$$

Demostració.

Per obtenir les equacions variacionals associades a ε i η expandirem Ω en els punts L_i per reescriure les equacions de moviment (2.4).

En efecte, l'expansió Ω en un entorn dels punts L_1 resulta:

$$\Omega = \Omega(a_i,b_i)+\Omega_x(a_i,b_i)\epsilon+\Omega_y(a_i,b_i)\eta+\frac{1}{2!}\Omega_{xx}(a_i,b_i)\epsilon^2+\Omega_{xy}(a_i,b_i)\epsilon\eta+\frac{1}{2!}\Omega_{yy}(a_i,b_i)\eta^2+O(3).$$

I per tant,

$$\Omega_x = \Omega_x(a_i,b_i) + \Omega_{xx}(a_i,b_i)\epsilon + \Omega_{xy}(a_i,b_i)\eta + O(2) = \Omega_{xx}(a_i,b_i)\epsilon + \Omega_{xy}(a_i,b_i)\eta + O(2),$$

$$\Omega_y = \Omega_y(a_i,b_i) + \Omega_{yx}(a_i,b_i)\epsilon + \Omega_{yy}(a_i,b_i)\eta + O(2) = \Omega_{xy}(a_i,b_i)\epsilon + \Omega_{yy}(a_i,b_i)\eta + O(2).$$

L'última igualtat en ambdues equacions ve donada perquè en els punts d'equilibri $\Omega_x(a,b) = \Omega_y(a,b) = 0$. Finalment, com que per hipòtesi $x_i = a_i + \varepsilon$ i $y = b_i + \eta$, podem reescriure les equacions de moviment com:

$$\begin{aligned} \ddot{\epsilon} - 2\dot{\eta} &= \Omega_{xx}(a,b)\epsilon + \Omega_{xy}(a,b)\eta + O(2), \\ \ddot{\eta} + 2\dot{\epsilon} &= \Omega_{xy}(a,b)\epsilon + \Omega_{yy}(a,b)\eta + O(2). \end{aligned} \tag{2.17}$$

Considerant els termes d'ordre 1 obtenim les equacions variacionals amb ϵ i η com a variacions. Per trobar el polinomi característic del sistema (2.17) primer l'escriurem com a un sistema de primer ordre. Escrivim:

$$x = (x_1, x_2, x_3, x_4)^T,$$

$$x_1 = \epsilon, \quad x_2 = \eta, \quad x_3 = \dot{\epsilon}, \quad x_4 = \dot{\eta}.$$

Podem escriure el sistema (2.17) com $\dot{x} = Ax$ on:

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \Omega_{xx} & \Omega_{xy} & 0 & 2 \\ \Omega_{xy} & \Omega_{xy} & 2 & 0 \end{pmatrix},$$

El polinomi característic el podem extreure mitjançant $\det(A - \lambda Id)$, en definitiva:

$$\begin{vmatrix} \lambda^2 - \Omega_{xx}^2 & -2\lambda - \Omega_{xy} \\ 2\lambda - \Omega_{xy} & \lambda^2 - \Omega_{yy}^2 \end{vmatrix} = 0, \quad \longrightarrow \quad \lambda^4 + (4 - \Omega_{xx} - \Omega_{yy})\lambda^2 + \Omega_{xx}\Omega_{yy} - \Omega_{xy}^2 = 0.$$

Tal com volíem veure. □

Observació 2.20. Notem que els punts col·lineals $L_{1,2,3}$ al tenir la coordenada y nul·la són del tipus $a_{1,2,3} = x_{1,2,3}$, $b = 0$ i en el cas dels punts equilaterals $L_{4,5}$ obtenim $a_{4,5} = \mu - \frac{1}{2}$, $b_{4,5} = \pm\frac{\sqrt{3}}{2}$.

2.4.2 Estabilitat lineal del punts col·lineals

A la secció anterior hem obtingut el polinomi característic que ens permetrà estudiar l'estabilitat dels punts. Abans, però, haurem d'estudiar que hi passa a les derivades segones d'Ω en els punts col·lineals.

Lema 2.21. *En els tres punts col·lineals la funció Ω compleix que:*

$$\Omega_{xy} = 0, \quad \Omega_{xx} > 0, \quad \Omega_{yy} < 0.$$

En particular, L_1, L_2 i L_3 són punts de sella.

Demostració.

Per a demostrar les propietats del lema partirem de les expressions de les derivades parcials d'Ω, les derivarem i avaluarem les expressions en els punts col·lineals. Per definició Ω_x i Ω_y tenim que:

$$\Omega_x = x - \frac{(1-\mu)(x-\mu)}{r_1^3} - \frac{\mu(x+1-\mu)}{r_2^3} = g(x,y),$$

$$\Omega_y = y\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right) = yf(x,y).$$

Si calculem les derivades en els punts col·lineals obtenim:

$$\Omega_{xx} = g_x(x,0) = 1 + \frac{2(1-\mu)(x-\mu)^2}{r_1^5} + \frac{2\mu(x+1-\mu)^2}{r_2^5},$$

$$\Omega_{yy} = f(x,0) = 1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3},$$

$$\Omega_{xy} = yf_x(x,0).$$

Clarament, $\Omega_{xy} = 0$ en tots els punts col·lineals. Per veure que $\Omega_{xx} > 0$ i que $\Omega_{yy} < 0$, caldrà jugar amb les expressions de les distàncies r_1 i r_2. Sigui x_1 l'abscissa del punt L_1, llavors, $x_1 - \mu = -r_1$ i $x_1 + 1 - \mu = -r_2$, així doncs:

$$\Omega_{yy} = g_x(x_1,0) = 1 + \frac{2(1-\mu)}{r_1^3} + \frac{2\mu}{r_2^3} > 0,$$

doncs tots els termes són positius. Com que en tots els punts col·lineals $\Omega_x = g(x_i,0) = 0$, tenim

$$g(x_i,0) = \mu - r_1 + \frac{1-\mu}{r_1^2} + \frac{\mu}{r_2^2} = 0 \longrightarrow \frac{1-\mu}{r_1^2} = -\mu + r_1 - + \frac{\mu}{r_2^2}.$$

Utilitzant l'última igualtat, podem escriure $f(x_1,0)$ de la següent forma:

$$f(x_1,0) = 1 - \frac{1-\mu}{r_1^2}\frac{1}{r_1} - \frac{\mu}{r_2^3} = 1 - \left(-\mu + r_1 - + \frac{\mu}{r_2^2}\right)\frac{1}{r_1} - \frac{\mu}{r_2^3},$$

$$f(x_1,0)=\mu\left(\frac{1}{r_1}+\frac{1}{r_1 r_2^2}-\frac{1}{r_2^2}\right)=\mu\left(\frac{1}{r_2}-\frac{1}{r_1}\right)\left(r_2-\frac{1}{r_2^2}\right).$$

Finalment,

$$f(x_1,0)=\frac{\mu}{r_1}\left(1-\frac{1}{r_2^3}\right).$$

Que $r_2<1$ implica que $\Omega_{yy}=f(x_1,0)<0$. Resta demostrar que L_1 és un punt de sella. Per veure-ho, calcularem el determinant de la hessiana avaluada a L_1:

$$H(L_1)=\begin{vmatrix}\Omega_{xx} & \Omega_{xy}\\ \Omega_{xy} & \Omega_{yy}\end{vmatrix}=g_x(x_1,0)f(x_1,0)<0.$$

En conclusió, L_1 és un punt de sella. Els casos L_2 i L_3 són pràcticament anàlegs al cas de L_1. □

Proposició 2.22. *Els punts col·lineals són linealment inestables.*

Demostració.

A conseqüència del lema anterior obtenim que en els punts col·lineals $\Omega_{xx}\Omega_{yy}<0$ i que $\Omega_{xy}^2=0$. Això ens permet reescriure el polinomi característic (2.16),

$$\lambda^4+(4-\Omega_{xx}-\Omega_{yy})\lambda^2+\Omega_{xx}\Omega_{yy}-\Omega_{xy}^2=0,$$

com:

$$\alpha^2+2\beta_1\alpha-\beta_2^2=0, \tag{2.18}$$

on:

$$\beta_1=2-\frac{\Omega_{xx}+\Omega_{yy}}{2},$$
$$\beta_2^2=-\Omega_{xx}\Omega_{yy}>0,$$
$$\lambda=\pm\alpha^{\frac{1}{2}}.$$

Per (2.18) tenim dues arrels de diferent signe.

$$\alpha_1=-\beta_1+(\beta_1^2+\beta_2^2)^{\frac{1}{2}}>0,$$
$$\alpha_2=-\beta_1-(\beta_1^2+\beta_2^2)^{\frac{1}{2}}<0.$$

Per tant, les arrels de (2.16) són:

$$\lambda_{1,2}=\pm\alpha_1^{\frac{1}{2}},$$
$$\lambda_{3,4}=\pm\alpha_2^{\frac{1}{2}}.$$

De les quals $\lambda_{1,2}$ són arrels reals i $\lambda_{3,4}$ són arrels imaginàries.

Sabem que les solucions del sistema d'equacions diferencials (en forma diagonal) són:

$$\epsilon=\sum_{i=1}^{4}=A_i e^{\lambda_i t},$$

$$\eta=\sum_{i=1}^{4}=B_i e^{\lambda_i t}.$$

I considerant l'arrel λ_1 com l'arrel real positiva, tindrem que per $t\to\infty$ aquest terme tendeix a infinit. En definitiva, els punts L_1, L_2 i L_3 són inestables. □

2.4.3 Estabilitat lineal dels punts equilaterals

L'estratègia per estudiar l'estabilitat lineal dels punts equilaterals serà molt semblant a la dels col·lineals. Començarem demostrant un lema que ens permetrà jugar amb el polinomi característic en el cas dels punts L_4 i L_5.

Lema 2.23. *Pels punts L_4 i L_5 tenim que:*

$$\Omega_{xx}(L_{4,5}) = \frac{3}{4}, \quad \Omega_{xy}(L_4) = +\frac{3\sqrt{3}}{2}(\mu - \frac{1}{2}),$$

$$\Omega_{xy}(L_5) = -\frac{3\sqrt{3}}{2}(\mu - \frac{1}{2}), \quad \Omega_{yy}(L_{4,5}) = \frac{9}{4}.$$

Demostració.

Recordem que teníem:

$$\Omega_x = x - \frac{(1-\mu)(x-\mu)}{r_1^3} - \frac{\mu(x+1-\mu)}{r_2^3} = g(x,y)$$

$$\Omega_y = y\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right) = yf(x,y)$$

A més a més, en els punts L_4 i L_5 tenim:

$$r_1 = r_2 = 1, \quad y = \pm\frac{\sqrt{3}}{2} \neq 0,$$

Que les distàncies r_1 i r_2 valguin totes dues 1 ens permet afirmar que $f(L_{4,5}) - g(L_{4,5}) = 0$, en efecte,

$$f(L_{4,5}) = -2\mu = g(L_{4,5}).$$

De tal manera que podem escriure les segones derivades d'Ω en funció de derivades de la funció g o bé f segons ens convingui. Si posem,

$$\Omega_{xx} = g_x, \quad \Omega_{xy} = g_y = yf_x, \quad \Omega_{yy} = yf_y + f.$$

Calculant les derivades de g i f i avaluant-les en els punts obtenim:

$$g_x(L_{4,5}) = \frac{3}{4}, \quad f_x(L_{4,5}) = 3(\mu - \frac{1}{2}), \quad f_y(L_{4,5}) = \pm 3\frac{\sqrt{3}}{2}.$$

Tal com volíem veure. □

Aplicant el lema anterior podem reescriure el polinomi característic (2.16),

$$\lambda^4 + (4 - \Omega_{xx} - \Omega_{yy})\lambda^2 + \Omega_{xx}\Omega_{yy} - \Omega_{xy}^2 = 0,$$

com,

$$\lambda^4 + \lambda^2 + \frac{27}{4}\mu(1-\mu) = 0. \tag{2.19}$$

Igual que en l'apartat anterior, mirarem si les arrels del polinomi característic són reals o imaginàries. Considerant una $\alpha = \lambda^2$, podem reescriure (2.19) per:

$$\alpha^2 + \alpha + \frac{27}{4}\mu(1-\mu) = 0.$$

On podem veure que les solucions són de la forma:

$$\alpha_{1,2} = \frac{1}{2}\left(-1 \pm [1 - 27\mu(1-\mu)]^{\frac{1}{2}}\right). \tag{2.20}$$

Veiem que tenim tres casos per avaluar segons el valor del discriminant $d = 1-27\mu(1-\mu)$, en conseqüència, del valor de la μ. Tenim que $d = 0$ quan $\mu_0 = \frac{27\pm\sqrt{27^2-108}}{54}$. Com que la $\mu \in (0, \frac{1}{2}]$ considerem només l'arrel $\mu_0 = 0.03852...$ que de fet, aquest valor crític s'anomena **massa de Routh**. A continuació, presentarem cada cas en forma de proposició.

Proposició 2.24. *En el cas $\mu \in (0, \mu_0)$ els punts L_4 i L_5 són linealment estables.*

Demostració.

En aquest rang, el valor del discriminat es troba entre 0 i 1. Per tant,

$$\alpha_1 \in \left(-\frac{1}{2}, 0\right], \quad \alpha_2 \in \left[-1, -\frac{1}{2}\right).$$

Els dos valors prenen valors negatius i, en conseqüència, les quatre arrels del polinomi característic són imaginàries.

Per a demostrar que tenir tots els valors propis imaginaris implica que el punt és estable en el nostre cas, haurem de fer ús del següent teorema.

Teorema 2.25. *Donat un sistema d'equacions diferencials $\dot{X}(t) = AX(t)$ on A té diferents parells de valors propis complexos, $\alpha_1 + i\beta_1$, $\alpha_1 - i\beta_1$, ... $\alpha_k + i\beta_k$, $\alpha_k - i\beta_k$. Sigui T la matriu tal que $T^{-1}AT$ és en forma canònica:*

$$T^{-1}AT = \begin{pmatrix} B_1 & & \\ & \ddots & \\ & & B_k \end{pmatrix}. \tag{2.21}$$

on

$$B_i = \begin{pmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha \end{pmatrix}.$$

Llavors la solució general del sistema és $TY(t)$, on

$$Y(t) = \begin{pmatrix} a_1 e^{\alpha_1 t}\cos\beta_1 t + b_1 e^{\alpha_1 t}\sin\beta_1 t \\ -a_1 e^{\alpha_1 t}\sin\beta_1 t + b_1 e^{\alpha_1 t}\cos\beta_1 t \\ \vdots \\ a_k e^{\alpha_k t}\cos\beta_k t + b_k e^{\alpha_k t}\sin\beta_k t \\ -a_k e^{\alpha_k t}\sin\beta_k t + b_k e^{\alpha_k t}\cos\beta_k t \end{pmatrix}. \tag{2.22}$$

Demostració.

La demostració es pot trobar en el capítol sis de [8]. □

Corol·lari 2.26. *En les hipòtesis del teorema anterior, sigui A la diferencial en el punt d'equilibri, si aquesta té tots els valors propis purament imaginaris (no nuls) llavors el punt és linealment estable.*

Demostració.

Sigui $c_{i,j}$ els components de la matriu T. Per definició T és invertible i, per tant, cada columna de la matriu conté, com a mínim, un valor no nul i totes les columnes són linealment independents. En definitiva, per a cada $j \in 1 \leq j \leq k$ podem trobar una i tal que $c_{i,2j-1} \neq 0$ o $c_{i,2j} \neq 0$ on $c_{i,2j-1} \neq \pm c_{i,2j}$.

Els valors propis no tenen part real tenim que $\alpha_i = 0\ \forall i$ en (2.22). També sabem que la solució general del sistema és $TY(t)$, i, en conseqüència, les seves components són

$$\sum_{i=1}^{k} \left[c_{i,2j-1}(a_1 \cos \beta_1 t + b_1 \sin \beta_1 t) + c_{i,2j}(-a_1 \cos \beta_1 t + b_1 \sin \beta_1 t) \right].$$

És clar que tots els termes estan acotats doncs el temps només apareix en cosinus i sinus. En definitiva, el punt és estable. □

Així doncs, la demostració de l'estabilitat de L_4 i L_5 és conseqüència directa del corol·lari. □

Proposició 2.27. *En el cas $\mu \in (\mu_0, \frac{1}{2})$ els punts L_4 i L_5 són linealment inestables.*

Demostració.

Sabem que el discriminat d del polinomi característic (2.20) és negatiu per $\mu \in (\mu_0, \frac{1}{2})$. A més a més, les arrels del polinomi característic són

$$\alpha_{1,2} = \frac{1}{2}(-1 \pm [d]^{\frac{1}{2}}).$$

Llavors, definint $\delta = [d]^{\frac{1}{2}}$

$$\lambda_1 = \frac{1}{\sqrt{2}}(-1 + i\delta)^{\frac{1}{2}} = \alpha_1 + i\beta_1, \quad \lambda_2 = -\frac{1}{\sqrt{2}}(-1 + i\delta)^{\frac{1}{2}} = \alpha_2 + i\beta_2,$$
$$\lambda_3 = \frac{1}{\sqrt{2}}(-1 - i\delta)^{\frac{1}{2}} = \alpha_3 + i\beta_3, \quad \lambda_4 = -\frac{1}{\sqrt{2}}(-1 - i\delta)^{\frac{1}{2}} = \alpha_4 + i\beta_4.$$

Amb les expressions explícites de les arrels veiem que per alguna $i \in \{1, 2, 3, 4\}$ la part real de λ_i serà positiva. Essent així els punts L_4 i L_5 inestables. □

Per últim, en el cas on el discriminant és nul obtindrem $\alpha_{1,2} = -\frac{1}{2}$ i com a solucions del polinomi característic (2.19) dues arrels dobles complexes.

$$\lambda_{1,3} = \frac{i}{\sqrt{2}}, \quad \lambda_{2,4} = -\frac{i}{\sqrt{2}}.$$

El fet d'obtenir dos valors propis dobles purament imaginaris farà la jacobiana no diagonalitzí i, en definitiva, resultarà en dues seccions no periòdiques. Els detalls es poden trobar a [5].

Hem vist doncs que en el cas que la μ sigui més petita que μ_0 llavors els punts equilaterals són linealment estables. Dit d'altre forma, si volem tenir estabilitat (lineal) en els punts L_4 i L_5, cal que la massa m_2 sigui menor aproximadament a un 4% de la massa de m_1. Tot i que sembla un valor molt baix, es poden trobar diversos exemples de sistemes amb aquesta restricció a l'espai. Per exemple, el sistema Terra-Lluna, Terra-Sol, Júpiter-Sol...

En definitiva, tots els punts de Lagrange són linealment inestables tret de L_4 i L_5 en el cas que la massa μ sigui menor al valor de la massa de Routh. No estudiarem l'estabilitat dels termes no lineals tot i que pels lectors més interessats es pot trobar novament a [5].

2.5 Corbes de velocitat zero

Un concepte clau per entendre el problema de tres cossos restringit són les corbes de velocitat zero. En darreres seccions vàrem veure que existia una constant, anomenada constant de Jacobi, la qual ens relacionava la velocitat amb la funció Ω. Precisament, en aquesta secció estudiarem a fons les conseqüències d'aquesta relació i farem un estudi de les diferents corbes de velocitat zero que existeixen segons el valor de la constant de Jacobi.

Havíem vist a l'observació (2.10) que podíem relacionar la velocitat de moviment de la partícula m_3 amb la constant de Jacobi per,

$$v^2 = 2\Omega - C. \tag{2.23}$$

És clar que la partícula m_3 mai tindrà el mòdul de la velocitat negativa i, de fet, podem delimitar la regió de moviment a partir de la condició $v^2 = 0$. Això ens portarà a la següent definició.

Definició 2.28. *Anomenem* ***corbes de velocitat zero*** *a les corbes que compleixen:*

$$\Omega(x, y) = \frac{C}{2}.$$

Observació 2.29. En aquestes corbes la partícula m_3 té velocitat nul·la i, per tant, aquesta no les pot creuar. En particular, les corbes de velocitat zero separen l'espai de regions possibles de moviment.

Efectivament, utilitzant l'equació $v^2 = 2\Omega - C$ notem que la condició $2\Omega = C$ ens divideix el espai en dues regions. Si $2\Omega < C$ el moviment no és possible doncs el mòdul de la velocitat seria negatiu. De tal manera que $2\Omega > C$ contindria tota la regió de moviment de la partícula m_3.

Seguidament, estudiarem algunes de les propietats més importants d'Ω que farem servir més tard per a classificar les diferents corbes de velocitat zero.

Proposició 2.30. *El valor mínim d'*Ω *és* $\frac{3}{2}$*. En particular, el valor mínim de la constant* C *és 3.*

Demostració.

Ens serà suficient provar que el valor mínim de 2Ω és 3, ja que per la definició de corbes de velocitat zero tenim $C = 2\Omega$.

Començarem reescrivint Ω d'una forma astuta. Afirmem que si $0 \leqq \mu \leqq 1$, $A, B \geqq 0$, llavors

$$A\mu + B(1-\mu) \geqq min(A, B).$$

En el cas $A \geqq B$,

$$A\mu + B(1-\mu) \geqq B\mu + B(1-\mu) = B = min(A, B).$$

Anàlogament per $B \geqq A$.

Per reescriure Ω, partirem de l'equació (2.23),

$$v^2 = 2\Omega - C.$$

Escrivint la velocitat en termes de coordenades bipolars ($v^2 = x^2 + y^2$), obtenim:

$$2\Omega = (1-\mu)(p_1^2 + 2p_1^{-1}) + \mu(p_2^2 + 2p_2^{-1}).$$

Ara ja podem utilitzar la desigualtat de l'inici,

$$2\Omega = (1-\mu)(p_1^2 + 2p_1^{-1}) + \mu(p_2^2 + 2p_2^{-1}) \geqq \min(p_1^2 + 2p_1^{-1}, p_2^2 + 2p_2^{-1}).$$

Finalment, el mínim de la funció $x^2 + 2x^{-1}$ és 3 i, per tant, també ho és de 2Ω. □

Observació 2.31. De fet, el valor mínim de la constant de Jacobi s'assoleix quan $p_1 = p_2 = 1$ (on s'assoleix el mínim de $x^2 + 2x^{-1}$). Per tant, en els punts L_4 i L_5.

La següent proposició ens permetrà separar l'estudi de les corbes de velocitat zero segons en quin rang es trobi el valor de la constant de Jacobi associada amb la corba.

Proposició 2.32. *Tenim quatre valors diferents d'*Ω *en els punts de Lagrange i segueixen el següent ordre:*

$$1.5 = \Omega(L_4) = \Omega(L_5) \leq \Omega(L_3) \leq \Omega(L_1) \leq \Omega(L_2) \leq 2.125.$$

En particular, això dona els següents valors de la constant de Jacobi,

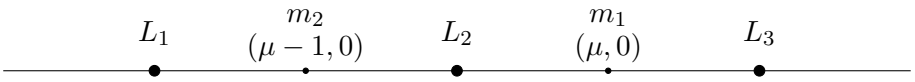
$$3 = C_4 = C_5 \leq C_3 \leq C_1 \leq C_2 \leq 4.25.$$

Demostració.

Els valors pels punts L_4 i L_5 ja s'ha provat a l'observació anterior. Pels punts L_1, L_2 i L_3 cal considerar que es troben l'eix de les x's segons el sistema de referència sinòdic. Sigui x_1, x_2 i x_3 les abscisses dels punts col·lineals L_1, L_2 i L_3 respectivament, volem veure que

$$1.5 \leq \Omega(x_3, 0) \leq \Omega(x_1, 0) \leq \Omega(x_2, 0) \leq 2.125.$$

Recordem que teníem el següent dibuix orientatiu pels punts col·lineals



Veurem a continuació que $\Omega(\mu + \alpha, 0) < \Omega(\mu - \alpha, 0)$ per $0 < \alpha < 1$. Tenim,

$$\begin{cases} \Omega(\mu+\alpha,0) = \frac{1}{2}(\mu+\alpha)^2 + \frac{\mu}{1+\alpha} + \frac{1-\mu}{\alpha} + \frac{\mu}{2}(1-\mu), \\ \Omega(\mu-\alpha,0) = \frac{1}{2}(\mu-\alpha)^2 + \frac{\mu}{1-\alpha} + \frac{1-\mu}{\alpha} + \frac{\mu}{2}(1-\mu). \end{cases}$$

Llavors,

$$\Omega(\mu-\alpha,0) - \Omega(\mu+\alpha,0) = \frac{2\mu}{1-\alpha^2}(\alpha^2 - \alpha + 1) > 0. \tag{2.24}$$

Perquè $(\alpha^2 - \alpha + 1) > 0$ per $\alpha \in (0, 1)$. Per continuar amb la demostració necessitarem els següents dos lemes,

Lema 2.33. *Siguin rang 1, rang 2 i rang 3 les regions de l'eix d'abscisses determinat pels punts col·lineals. Els mínims d'*Ω *a cada regió són els respectius punts L_1, L_2 i L_3.*

Demostració.

Considerant la funció Ω:

$$\Omega(x,0) = \frac{1}{2}x^2 + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{\mu}{2}(1-\mu).$$

Tenim,

$$\begin{cases} \frac{d\Omega(x,0)}{dx} = x - \frac{1-\mu}{r_1^2}\frac{dr_1}{dx} - \frac{\mu}{r_2^2}\frac{dr_2}{dx}, \\ \frac{d^2\Omega(x,0)}{dx^2} = 1 + \frac{2(1-\mu)}{r_1^3}\left(\frac{dr_1}{dx}\right) + \frac{2\mu}{r_2^3}\left(\frac{dr_2}{dx}\right)^2 > 1. \end{cases} \tag{2.25}$$

Com la segona derivada és positiva $\forall x$, serà suficient veure que en els punts col·lineals és on s'anul·la la primera derivada.

Ens situem en el rang 1 (on es troba L_1). Tenim $r_1 = \mu - x$, $r_2 = \mu - x - 1$ i, en conseqüència, $\frac{dr_1}{dx} = \frac{dr_2}{dx} = -1$. Aplicant-ho a la primera derivada,

$$\frac{d\Omega(x,0)}{dx} = x + \frac{1-\mu}{r_1^2} + \frac{\mu}{r_2^2}. \tag{2.26}$$

Idèntica a l'equació (2.8). La demostració pels rangs 2 i 3 té un desenvolupament similar. En tots tres casos s'obté les equacions de cinquè grau que donen lloc als punts col·lineals ja estudiats anteriorment. □

Lema 2.34. *Per $\mu \in (0, \frac{1}{2})$, tenim:*

$$r_1(L_3) > r_1(L_2).$$

Demostració.

La demostració segueix propietats de monotonia de les distàncies. En concret es pot trobar a [10]. □

Continuant amb la demostració de la proposició, tenim pel primer lema podem afirmar que $\Omega(x_3,0) < \Omega(\mu+\alpha,0)$ per $0 < \alpha < x_3 - \mu$. També pel segon lema, $x_3 - \mu > \mu - x_2$. Per tant, si $\mu - \alpha = x_2$, tenim per (2.24) que $\Omega(x_3,0) < \Omega(x_2,0)$. De forma semblant es pot veure $\Omega(x_1) < \Omega(x_2,0)$. □

Fixem-nos que podem utilitzar els diferents valors de la constant de Jacobi descrits a la proposició (2.32) per a estudiar les corbes de velocitat zero. En efecte, dividirem els possibles valors de la constant de Jacobi en els intervals:

$$[3, C_3], \quad (C_3, C_1], \quad (C_1, C_2], \quad (C_2, +\infty).$$

En cada cas trobarem un dibuix de la forma general de les corbes de velocitat zero on destacarem les seves principals característiques. [4]

2.5.1 Regions de moviment $C_2 < C$

Estudiarem el primer cas, on la constant de Jacobi pren els valors més grans possibles. Per tal de fer-ho, analitzarem la funció Ω amb $\mu \in (0, \frac{1}{2}]$.

[4] Tots els dibuixos de corbes de velocitat zero han estat generals pel programa *Corbes de velocitat zero* en llenguatge C descrit als annexos **B**.

Primer haurem de reescriure la funció Ω utilitzant les distàncies, recordem que

$$\Omega = \frac{1}{2}\left(x^2+y^2\right) + \frac{\mu}{r_2} + \frac{1-\mu}{r_1} + \frac{1}{2}\mu\left(1-\mu\right).$$

Utilitzant que les distàncies $r_1^2 = (x-\mu)^2 + y^2$ i $r_2^2 = (x+(1-\mu))^2 + y^2$, llavors:

$$(1-\mu)r_1^2 = x^2 + y^2 - 2\mu + 3\mu^2 - \mu^3 - \mu x^2 - \mu y^2,$$

$$\mu r_2^2 = \mu x^2 + \mu y^2 + 3\mu - 4\mu^2 + \mu^3.$$

De tal manera que la suma ens dona $x^2 + y^2 + \mu - \mu^2$. Per tant, podem reescriure Ω de la següent forma:

$$\Omega = \frac{1}{2}\left[(1-\mu)r_1^2 + \mu r_2^2\right] + \frac{\mu}{r_2} + \frac{1-\mu}{r_1}. \tag{2.27}$$

Per la forma d'Ω diferenciem tres casos on podem obtenir un valor gran, quan r_1 tendeix a zero, quan r_2 tendeix a zero o bé quan les dues distàncies prenen valors molt grans. Veurem que cada cas és, de fet, una corba de velocitat zero.

i) **Oval de velocitat zero al voltant de la primària de major massa.**

Considerem que la distància $r_1 \to 0$, llavors $r_2 \to 1$. Obtenim doncs:

$$\Omega = \frac{1}{2}\mu + \mu + \frac{1-\mu}{0}.$$

Per tant, el valor dominant és $\frac{1-\mu}{0}$ i la nostra constant $C = 2(1-\mu)/r_1$. Com a conseqüència, obtenim que les corbes de velocitat zero són ovals al voltant de m_1. Com més gran sigui el valor de C més petit serà r_1, ja que $\frac{d\Omega}{dr_1} \cong -\frac{1-\mu}{r_1^2} < 0$.

Proposició 2.35. *En aquest cas el moviment de la partícula estarà concentrat dins de l'oval de velocitat zero.*

Demostració.

Sigui v_o i $r_1 = r_0$ les condicions inicials tals que el corresponent valor de la Constant de Jacobi $C_0 \cong \frac{2(1-\mu)}{r_0} - v_0^2$. Tenim que la corba de velocitat zero per aquest valor $C = C_0$ és aproximadament un cercle de radi $2(1-\mu)/C_0 = r_2$. Com que $r_2 > r_0$,

$$v_0^2 = 2(1-\mu)(1/r_0 - 1/r_2) > 0.$$

Escollint un altre punt a distància r_0' de la massa primària, la seva velocitat v_0' és:

$$v_0^{2'} = 2(1-\mu)/r_0' - C_0 > 0.$$

O bé:

$$v_0^{2'} = 2(1-\mu)(1/r_0' - 1/r_2) > 0.$$

Per tant, sempre que la partícula sigui dins del cercle de velocitat zero, $r_0' < r_2$, v_0' serà un valor real i el moviment serà possible. En conclusió, la partícula no pot creuar l'espai delimitat per la corba de velocitat zero. □

ii) **Oval de velocitat zero al voltant de la primària de menor massa.**

Quan tenim $r_2 \ll 1$ i $r_1 \cong 1$, les corbes de velocitat zero són ovals al voltant de la massa petita. En aquest cas el radi de les corbes de velocitat zero és aproximadament:

$$r_2 \cong 2\mu/C.$$

Respecte al radi de l'anterior cas, si considerem una $C > 0$ prou gran, tindrem que el radi r_1 és més gran que el radi r_2. De fet,

$$0 < r_1 - r_2 \cong 2/C < \frac{2}{C} < \frac{2}{3}.$$

L'última desigualtat és conseqüència del fet que el mínim del valor de C sigui 3.

iii) **Ovals de velocitat zero al voltant de les dues masses primàries.**

En el cas que els dos radis tinguin valors prou grans ($r_1, r_2 \geqq 1$), podem obtenir valors de C relativament grans. De fet, si escollim $r_1 = r_2 = r$, tenim que la funció Ω es té la forma $\Omega = \frac{r^2}{2} + \frac{1}{r} \cong r^2$.

Amb distàncies grans els efectes de la gravetat són ínfims comparats amb l'efecte de la força centrífuga (d'on obtenim la r^2). La integral de Jacobi és

$$v^2 = r^2 - C,$$

i les corbes de velocitat zero són definides per

$$C^{\frac{1}{2}} \cong r_z.$$

Que són aproximadament cercles.

Observació 2.36. El radi obtingut serà sempre més gran que els radis anteriors. Per $C > 3$, $r_1 = 2(1-\mu)/C < C^{\frac{1}{2}} = r_z$. Com més gran sigui C, més gran seran els ovals de velocitat zero i, per tant, el moviment serà possible fora d'aquestes corbes (perquè r_z contindrà els ovals).

El primer valor crític que trobarem en reduir la constant de Jacobi serà $C = 2\Omega(L_2)$. En aquest cas obtenim una figura semblant a un vuit girat. A mesura que decreix el valor de la C, els ovals interiors s'amplien fins a trobar-se en el punt L_2 tot i que dintre d'aquest rang encara continuen deixant els altres punts col·lineals fora.

Les interseccions amb l'eix x's a la part dreta de la massa m_1 es poden trobar mitjançant l'expressió d'Ω (2.27),

$$C = 2\Omega = r_1^2 + \mu(r_2^2 - r_1^2) + \frac{2\mu}{r_2} + \frac{2(1-\mu)}{r_1}. \tag{2.28}$$

Com que estem a la dreta de la massa m_1, podem substituir $r_2 = r_1 + 1$,

$$C = r_1^2 + \mu(2r_1 + 1) + \frac{2\mu}{r_1 + 1} + \frac{2(1-\mu)}{r_1}. \tag{2.29}$$

Observació 2.37. Sigui x_3 l'abscissa de L_3. Pel valor de C sabem que L_3 separa els ovals interiors amb els exteriors, per tant, una arrel de (2.29) haurà de ser més petita que $x_3 - \mu$ i l'altre més gran.

També podem veure els talls de les corbes sobre l'eix de les x's a la part esquerra de m_2. En aquest cas tindríem $r_1 = 1 + r_2$ i substituint a (2.28) obtindríem,

$$C = (1 + r_2)^2 - \mu(2r_2 + 1) + \frac{\mu}{r_2} + \frac{2(1-\mu)}{1+r_2}.$$

Aquest cas torna a ser anàleg a l'anterior, però el nostre punt de separació entre les corbes interiors i exterior serà L_1.

La següent figura il·lustra la forma de les corbes de velocitat zero en el cas $C = 4 > C_2$ per $\mu = 0.2$. L'àrea ombrejada correspon a l'àrea possible de moviment.

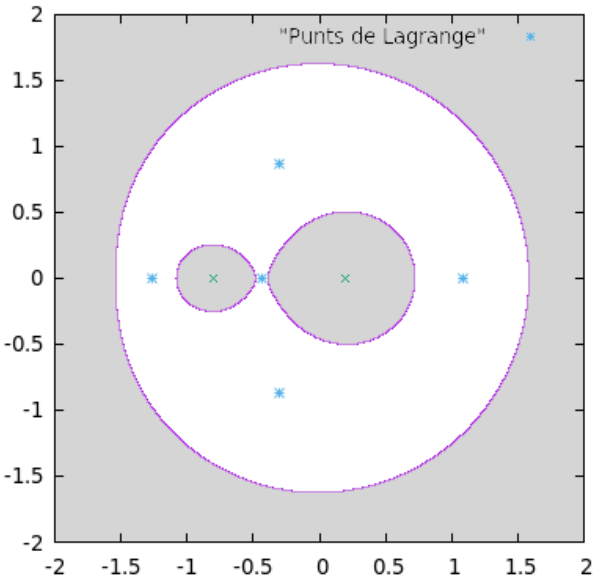


Figura 5: Corbes de velocitat zero per $\mu = 0.2$ i $C = 4$.

2.5.2 Regions de moviment $C_1 < C \leq C_2$

Entrem ara en el segon cas, on ens trobarem amb les corbes en forma de "pera". El valor de la C quan arriba al punt crític $C = C_2$ la forma de vuit toca el punt L_2 com havíem esmentat en el cas anterior. Al rebaixar el valor de la constant de Jacobi el vuit es transforma en dues branques. La interior té forma de pera, envoltant les dues masses primàries i el punt L_2. Per altra part, l'exterior envolta els punts L_3 i en el valor crític, $C = C_1$ conté el punt L_1.

A la figura 6 ambdues corbes tenen $\mu = 0.3$. En el cas de la corba de color lila la constant de Jacobi val $C = 4.5$ i verda $C = C_2 = 4.13$. Es veu clarament com els ovals interiors s'ajunten en un punt, essent aquest L_2.

2.5.3 Regions de moviment $C_3 < C \leq C_1$

Quan el valor de la constant de Jacobi es troba en aquest rang, passem a tenir només una branca (amb forma de ferradura) que conté els punts L_3, L_4 i L_5. Les interseccions amb l'eix de les x's s'apropen al punt L_3 quan $C \to C_3$ fins a arribar a coincidir en el valor crític.

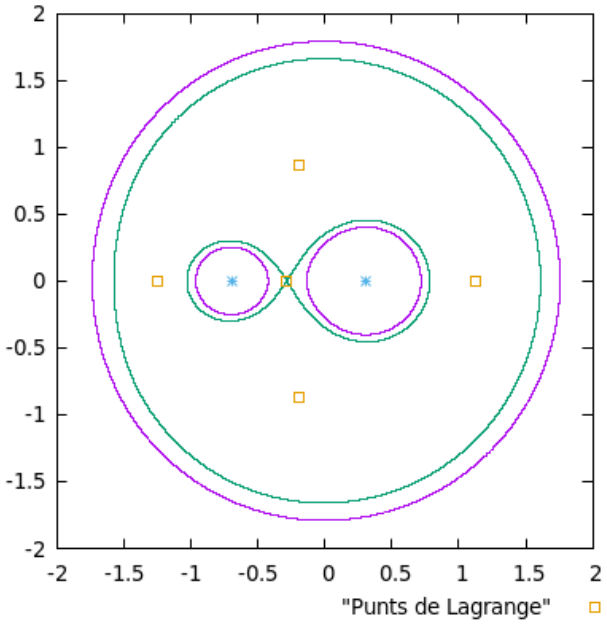


Figura 6: Comparativa entre corbes de velocitat zero. La corba verda té per a $C = C_2$.

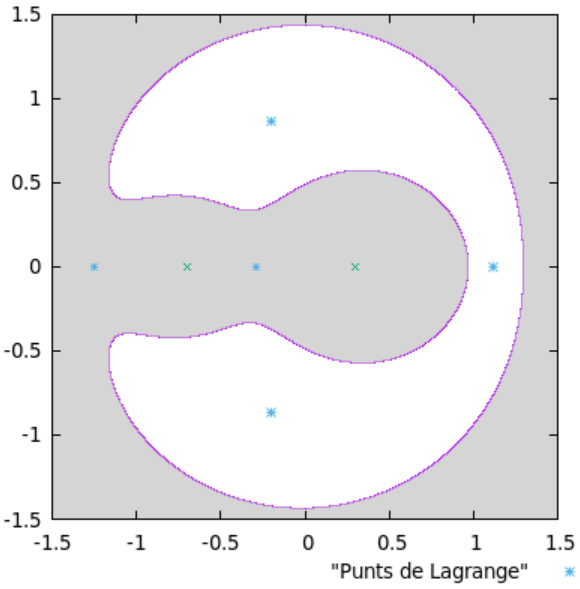


Figura 7: Corba de velocitat zero per $\mu = 0.3$ i $C = 3.6$.

2.5.4 Regions de moviment $3 = C_5 = C_4 \leq C \leq C_3$

En l'últim cas de la nostra discussió ens trobem que les corbes de velocitat zero a mesura que reduïm el seu valor (respecte C_3), se separen en dues branques on cada una envolta un punt equilateral.

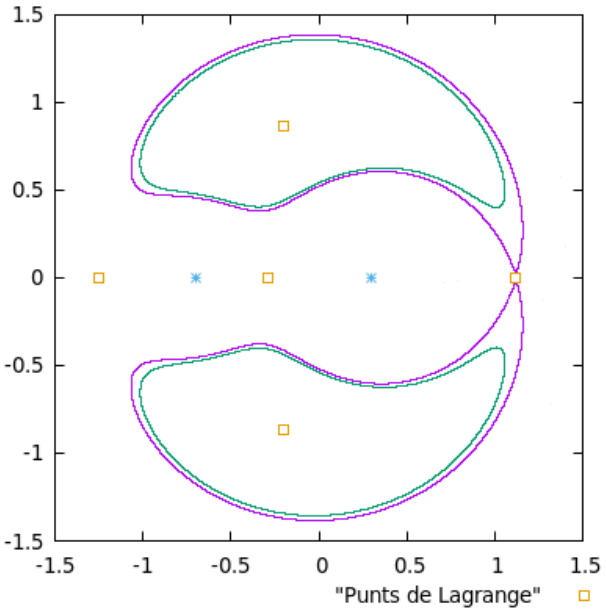


Figura 8: Corbes de velocitat zero amb $\mu = 0.3$, $C = 3.50103$ (lila) i $C = 3.45$ (verda).

2.5.5 Exemple sistema Terra-Lluna

Per finalitzar l'estudi de les corbes de velocitat zero, veurem les diferents corbes que existeixen en el sistema Terra-Lluna. Pintarem les corbes de diversos colors per diferenciar els diferents valors de la constant de Jacobi.

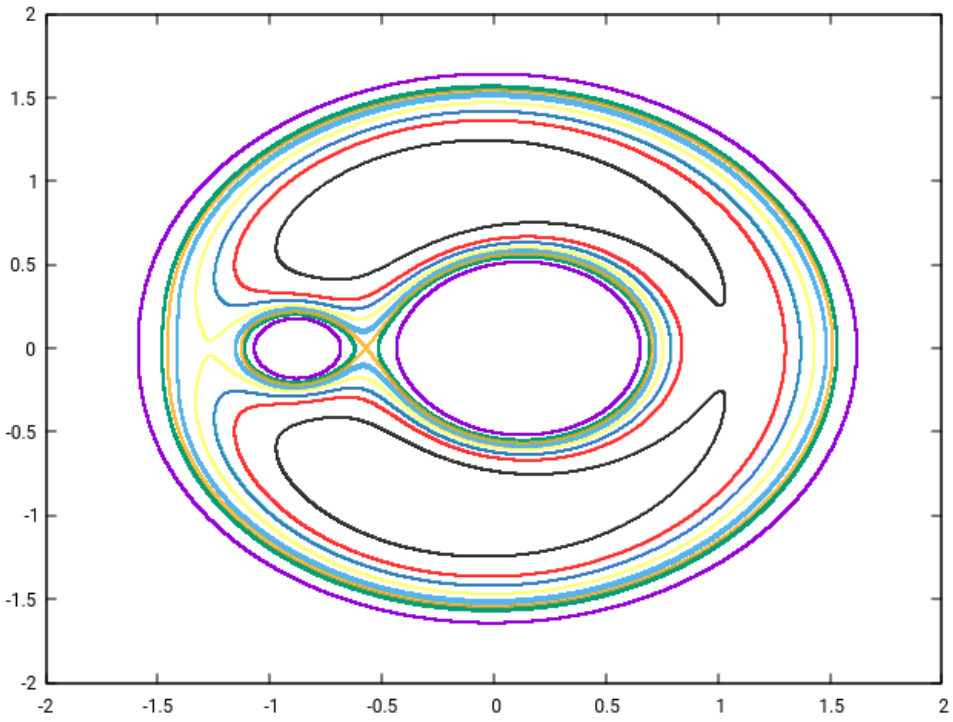


Figura 9: Corbes de velocitat zero del sistema Terra-Lluna.

3 Problema bicircular

Una simplificació del problema de quatre cossos és el problema bicircular (**PCB**). Aquest model, tot i no ser coherent amb la llei de Gravitació universal, aproxima molt bé el moviment d'una partícula en els sistemes de tres grans masses. A més a més, el model es basa en gran part en el moviment de tres cossos, el qual ja hem estudiat a fons en l'anterior capítol. Per totes aquestes raons i més que en veurem, ens dedicarem a estudiar el model bicircular, en particular, dedicarem aquest capítol a obtenir les equacions de moviment.

Suposem doncs que tenim quatre masses m_1, m_2, m_3 i m_4 tals que $m_4 \ll m_1, m_2, m_3$, de manera que podem considerar que l'acció de les forces gravitacionals derivades de la massa m_4 sobre les altres masses és negligible. Suposem també que m_1 i m_2 giren al voltant del seu centre de masses m_c amb òrbites circulars i que alhora m_c i m_3 giren també amb òrbites circulars al voltant del centre de masses de m_c (amb massa $m_1 + m_2$) i m_3.

Definició 3.1. *En les hipòtesi anteriors, anomenen* ***problema bicircular*** *a l'estudi del moviment de la partícula* m_4.

Observació 3.2. El model bicircular es pot pensar com dos problemes de tres cossos simultanis.

En efecte, suposem que estem estudiant el moviment d'una partícula p en el sistema Terra-Lluna-Sol. Sabem que la Terra i la Lluna tenen un centre de masses m_{TL} i aquestes giren en òrbites circulars al seu voltant. Alhora podem considerar m_{TL} i el Sol com l'altre problema de tres cossos restringit planar i circular.

Observació 3.3. A causa del moviment descrit pels tres cossos amb massa, el PCB no compleix la llei de conservació de l'energia i, per tant, tampoc segueix una coherència amb la llei de Gravitació Universal de Newton. En definitiva, podem afirmar que el model bicircular no és dinàmicament coherent. Tanmateix, com ja hem esmentat anteriorment, utilitzar el model bicircular garanteix una bona precisió quan volem aproximar el moviment d'una partícula amb massa negligible dins d'un sistema on hi ha tres masses significativament grans.

3.1 Equacions de moviment del problema bicircular

L'objectiu d'aquesta secció serà obtenir unes equacions de moviment de la partícula m_4 del problema bicircular, per a posteriorment, buscar-ne òrbites periòdiques. Així doncs, plantejarem primer el sistema de referència, les unitats i realitzarem diverses modificacions a les equacions fins a trobar-ne les desitjades.

Al llarg de tota la secció treballarem amb el sistema Terra-Lluna-Sol, on denotarem la massa de la Terra M_T, la massa de la Lluna per M_L i la massa del Sol per M_S. Escollirem com a unitat de massa $M_T + M_L$. En particular:

$$\mu = \frac{M_L}{M_T + M_L}, \quad 1 - \mu = \frac{M_T}{M_T + M_L}, \quad m_S = \frac{M_S}{M_T + M_L}.$$

El valor numèric aproximat és:

$$\mu \approx 0.012505819187, \quad m_S \approx 328900.549999.$$

On la massa total del sistema és $M = 1 + m_S$.

Anotarem la velocitat angular del Sol respecte al sistema Terra-Lluna com el moviment mig n_S, que prendrà el valor:

$$n_S \approx 0.0748040144814.$$

De tal manera que la freqüència del moviment del Sol respecte el sistema Terra-Lluna és:

$$\omega_S = 1 - n_S \approx 0.925195985518.$$

Podem calcular la distància mitja a_S entre m_{TL} i el Sol,

$$a_S = \left(\frac{1+m_S}{n_S^2}\right)^{\frac{1}{3}} \approx 388.8111430233.$$

Plantejarem ara el sistema de coordenades centrant-nos primer en les masses de la Terra i la Lluna i, posteriorment, realitzarem una translació tot considerant el centre de masses de la Terra-Lluna-Sol.

Escollirem com a origen el centre de masses de la Terra i la Lluna m_{TL} i com a eix de les x's la línia que uneix m_T i m_L. L'eix de les z's estarà orientat seguint la direcció del vector de moviment angular dels dos primaris respecte al centre de masses (essent perpendicular al pla del moviment). Finalment, escollirem l'eix de les y's de forma que el sistema de referència obtingut sigui ortogonal i definit positivament.

En definitiva, el sistema de referència (X, Y, Z), amb dependència respecte al temps t resulta:

$$\begin{aligned} X_T = \mu\cos t, \quad X_L = (1-\mu)\cos t, \quad X_S = a_S\cos(n_S t), \\ Y_T = \mu\sin t, \quad Y_L = (1-\mu)\sin t, \quad Y_S = a_S\sin(n_S t), \\ Z_T = 0, \quad Z_L = 0, \quad Z_S = 0. \end{aligned} \tag{3.1}$$

Observació 3.4. A $t = 0$, m_T té coordenades $(\mu, 0, 0)$, m_L $(1-\mu, 0, 0)$ i m_S $(a_S, 0, 0)$.

Ara voldrem modificar l'origen situant-lo al centre de masses dels tres cossos O. A causa del moviment circular del Sol respecte del sistema Terra-Lluna, el punt O dependrà del moviment mig n_S i de la distància mitja a_S. En definitiva, haurem d'aplicar la translació al centre de masses del sistema de coordenades al punt:

$$O = \left(-\frac{m_S}{M}a_S\cos(n_S t), -\frac{m_S}{M}a_S\sin(n_S t), 0\right).$$

El nou sistema de coordenades (U, V, W) aplicada la translació és:

$$\begin{cases} U = X - \frac{m_S}{M}a_S\cos(n_S t), \\ V = Y - \frac{m_S}{M}a_S\sin(n_S t), \\ W = Z. \end{cases} \tag{3.2}$$

Proposició 3.5. *Les derivades segones de les equacions (3.2) queden com:*

$$\begin{cases} \ddot{X} = \ddot{U} - \frac{m_S}{a_S^2}\cos(n_S t), \\ \ddot{Y} = \ddot{V} - \frac{m_S}{a_S^2}\sin(n_S t), \\ \ddot{Z} = \ddot{W}. \end{cases} \tag{3.3}$$

Demostració.

Prenem les equacions (3.2) i derivem dos cops respecte al temps,

$$\begin{cases} \ddot{X} = \ddot{U} - \frac{m_S}{M} n_S^2 a_S \cos(n_S t), \\ \ddot{Y} = \ddot{V} - \frac{m_S}{M} n_S^2 a_S \sin(n_S t), \\ \ddot{Z} = \ddot{W}. \end{cases}$$

La tercera llei de Kepler implica que:

$$n_S^2 = \frac{M}{a_S^3} \to \frac{m_S a_S n_S^2}{1+m_S} = \frac{m_S}{a_S^2}.$$

Finalment, apliquem la igualtat i obtenim les equacions desitjades. □

Proposició 3.6. *La posició d'una massa infinitesimal influenciada pels tres cossos m_T, m_L i m_S vindrà descrita per les equacions,*

$$\begin{cases} \ddot{X} = -\frac{(X-X_T)(1-\mu)}{r_{PT}^3} - \frac{(X-X_L)\mu}{r_{PL}^3} - \frac{(X-X_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\cos(n_S t), \\ \ddot{Y} = -\frac{(Y-Y_T)(1-\mu)}{r_{PT}^3} - \frac{(Y-Y_L)\mu}{r_{PL}^3} - \frac{(Y-Y_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\sin(n_S t), \\ \ddot{Z} = -\frac{Z(1-\mu)}{r_{PT}^3} - \frac{Z\mu}{r_{PL}^3} - \frac{Zm_S}{r_{PS}^3}. \end{cases} \tag{3.4}$$

On r_{PT}, r_{PL} i r_{PS} denoten la distància euclidiana del punt a la Terra, Lluna i Sol respectivament.

Demostració.

Partint de les equacions (3.3), aplicarem la llei de la Gravitació Universal de Newton a la partícula i calcularem les derivades parcials de la força aplicada a la partícula. Recordem que les coordenades de les masses m_T, m_L i m_S són $(X_T, Y_T, 0)$, $(X_L, Y_L, 0)$ i $(X_S, Y_S, 0)$ respectivament.

Cal recordar que r_{PT}, r_{PL} i r_{PS} són les distàncies euclidianes entre la partícula i les masses m_T, m_L i m_S. Més explícitament,

$$r_{PT}^2 = (X-X_T)^2 + (Y-Y_T)^2 + Z^2,$$
$$r_{PL}^2 = (X-X_L)^2 + (Y-Y_L)^2 + Z^2,$$
$$r_{PS}^2 = (X-X_S)^2 + (Y-Y_S)^2 + Z^2.$$

□

Ara el nostre objectiu serà canviar novament les coordenades per no només simplificar el model sinó poder treballar el PBC com una pertorbació (que no petita) del problema restringit de tres cossos. Per tal d'assolir el canvi desitjat farem ús de les coordenades sinòdiques que limitaran el moviment de la Terra i la Lluna dins l'eix de les x's.

Proposició 3.7. *Utilitzant el canvi de coordenades*

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{3.5}$$

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.6}$$

Les equacions (3.4) queden com:

$$\begin{cases} \ddot{x} = 2\dot{y} + x - \frac{1-\mu}{r_{PT}^3}(x-\mu) - \frac{\mu}{r_{PL}^3}(x-\mu+1) - \frac{m_S}{r_{PS}^3}(x - a_S\cos(t-n_S t)) - \frac{m_S}{a_S^2}\cos(t-n_S t), \\ \ddot{y} = -2\dot{x} + y - \frac{1-\mu}{r_{PT}^3}y - \frac{\mu}{r_{PL}^3}y - \frac{m_S}{r_{PS}^3}(y + a_S\sin(t-n_S t)) + \frac{m_S}{a_S^2}\sin(t-n_S t), \\ \ddot{z} = -\frac{1-\mu}{r_{PT}^3}z - \frac{\mu}{r_{PL}^3} - \frac{m_S}{r_{PS}^3}z. \end{cases} \tag{3.7}$$

Demostració.

Primer de tot, veiem que el canvi a coordenades sinòdiques no afecta la variable Z. Si derivem dos cops respecte al temps t el canvi de coordenades $X = X(x,y)$, $Y = Y(x,y)$ obtenim,

$$\begin{cases} \ddot{X} = \ddot{x}\cos t - 2\dot{x}\sin t - x\cos t - \ddot{y}\sin t - 2\dot{y}\cos t + y\sin t, \\ \ddot{Y} = \ddot{x}\sin t + 2\dot{x}\cos t - x\sin t + \ddot{y}\cos t - 2\dot{y}\sin t - y\cos t. \end{cases} \tag{3.8}$$

Que escrit de forma matricial és:

$$\begin{pmatrix} \ddot{X} \\ \ddot{Y} \end{pmatrix} = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix}\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} + \begin{pmatrix} -2\sin t & -2\cos t \\ 2\cos t & -2\sin t \end{pmatrix}\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} + \begin{pmatrix} -\cos t & \sin t \\ -\sin t & -\cos t \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.9}$$

Aïllant el vector que conté les derivades segones de l'equació (3.9) obtenim:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = -\begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} -2\sin t & -2\cos t \\ 2\cos t & -2\sin t \end{pmatrix}\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}$$

$$-\begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} -\cos t & \sin t \\ -\sin t & -\cos t \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} \ddot{X} \\ \ddot{Y} \end{pmatrix}.$$

Multiplicant les matrius tenim:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ -2 & 0 \end{pmatrix}\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} \ddot{X} \\ \ddot{Y} \end{pmatrix}. \tag{3.10}$$

Per veure el valor de l'últim terme farem ús de les equacions (3.4):

$$\begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} \ddot{X} \\ \ddot{Y} \end{pmatrix}$$

$$= \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} -\frac{(X-X_T)(1-\mu)}{r_{PT}^3} - \frac{(X-X_L)\mu}{r_{PL}^3} - \frac{(X-X_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\cos(n_S t) \\ -\frac{(Y-Y_T)(1-\mu)}{r_{PT}^3} - \frac{(Y-Y_L)\mu}{r_{PL}^3} - \frac{(Y-Y_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\sin(n_S t) \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{\cos t(X-X_T)(1-\mu)}{r_{PT}^3} - \frac{\cos t(X-X_L)\mu}{r_{PL}^3} - \frac{\cos t(X-X_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\cos(n_S t)\cos t \\ \frac{\sin t(X-X_T)(1-\mu)}{r_{PT}^3} + \frac{\sin t(X-X_L)\mu}{r_{PL}^3} + \frac{\sin t(X-X_S)m_S}{r_{PS}^3} + \frac{m_S}{a_S^2}\cos(n_S t)\sin t \\ -\frac{\sin t(Y-Y_T)(1-\mu)}{r_{PT}^3} - \frac{\sin t(Y-Y_L)\mu}{r_{PL}^3} - \frac{\sin t(Y-Y_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\sin(n_S t)\sin t \\ -\frac{\cos t(Y-Y_T)(1-\mu)}{r_{PT}^3} - \frac{\cos t(Y-Y_L)\mu}{r_{PL}^3} - \frac{\cos t(Y-Y_S)m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\sin(n_S t)\cos t \end{pmatrix}.$$

Aplicaren ara a les equacions les substitucions dels termes dependents del sistema de referència amb les equacions (3.1) i (3.5). Veure'm que amb:

$$\begin{cases} (X-X_T) = x\cos t - y\sin t - \mu\cos t, \\ (X-X_L) = x\cos t - y\sin t - (1-\mu)\cos t, \\ (X-X_S) = x\cos t - y\sin t - a_S\cos(n_S t), \\ (Y-Y_T) = x\sin t + y\cos t - \mu\sin t, \\ (Y-Y_L) = x\sin t + y\cos t - (1-\mu)\sin t, \\ (Y-Y_S) = x\sin t + y\cos t - a_S\sin(n_S t), \end{cases}$$

i simplificant els termes amb $\cos^2 t + \sin^2 t = 1$, ens resulta:

$$= \begin{pmatrix} -\frac{(x-\mu)(1-\mu)}{r_{PT}^3} - \frac{(x-\mu+1)\mu}{r_{PL}^3} - \frac{(x-a_S\cos(n_S t)\cos t - a_S\sin(n_S t)\sin t)m_S}{r_{PS}^3} \\ -\frac{y(1-\mu)}{r_{PT}^3} + \frac{y\mu}{r_{PL}^3} + \frac{(y+a_S\cos(n_S t)\sin t - a_S\sin(n_S t)|\cos t)m_S}{r_{PS}^3} \\ -\frac{m_S}{a_S^2}\cos(n_S t)\cos t - \frac{m_S}{a_S^2}\sin(n_S t)\sin t \\ +\frac{m_S}{a_S^2}\cos(n_S t)\sin t - \frac{m_S}{a_S^2}\sin(n_S t)\cos t \end{pmatrix}.$$

Ara, per acabar d'escriure les equacions tal com volem, ens caldrà aplicar la següent identitat trigonomètrica:

$$\begin{cases} \cos(t-n_S t) = \cos(n_S t)\cos t + \sin(n_S t)\sin t, \\ \sin(t-n_S t) = \cos(n_S t)\sin t - \sin(n_S t)\cos t. \end{cases} \tag{3.11}$$

Així doncs,

$$\begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}\begin{pmatrix} \ddot{X} \\ \ddot{Y} \end{pmatrix} = \begin{pmatrix} -\frac{(x-\mu)(1-\mu)}{r_{PT}^3} - \frac{(x-\mu+1)\mu}{r_{PL}^3} - \frac{(x-a_S\cos(t-n_S t))m_S}{r_{PS}^3} - \frac{m_S}{a_S^2}\cos(t-n_S t) \\ -\frac{y(1-\mu)}{r_{PT}^3} + \frac{y\mu}{r_{PL}^3} + \frac{(y+a_S\sin(t-n_S t))m_S}{r_{PS}^3} + \frac{m_S}{a_S^2}\sin(t-n_S t) \end{pmatrix}.$$

Finalment, substituint la darrera equació a (3.10) obtenim el resultat. □

Observació 3.8. Després del canvi de coordenades, r_{PT}, r_{PL} i r_{PS} prenen els valors següents,

$$\begin{aligned} r_{PT}^2 &= (X-X_T)^2 + (Y-Y_T)^2 + Z^2 = (x-\mu)^2 + y^2 + z^2, \\ r_{PL}^2 &= (X-X_L)^2 + (Y-Y_L)^2 + Z^2 = (x-\mu+1)^2 + y^2 + z^2, \\ r_{PS}^2 &= (X-X_S)^2 + (Y-Y_S)^2 + Z^2 = (x-x_S)^2 + (y-y_S)^2 + z^2, \end{aligned}$$

on $x_S = a_S\cos(t-n_S t)$ i $y_S = -a_S\sin(t-n_S t)$.

Si realitzem un petit canvi a les equacions de la proposició anterior, podrem obtenir un hamiltonià que defineixi el model bicircular. El presentarem escrit en la següent proposició.

Proposició 3.9. *El model bicircular té com a equacions de moviment,*

$$\begin{cases} \dot{x} = p_x + y, \\ \dot{y} = p_y - x, \\ \dot{z} = p_z, \\ \dot{p_x} = p_y - \frac{(1-\mu)}{r_{PT}^3}(x-\mu) - \frac{\mu}{r_{PL}^3}(x-\mu+1) + \frac{m_S}{r_{PS}^3}(x-x_S) - \frac{m_S}{a_S^3}x_S, \\ \dot{p_y} = -p_x - \frac{(1-\mu)y}{r_{PT}^3} - \frac{\mu y}{r_{PL}^3} - \frac{m_S}{r_{PS}^3}(y-y_S) - \frac{m_S}{a_S^3}y_S, \\ \dot{p_z} = -\left(\frac{1-\mu}{r_{PT}^3} + \frac{\mu}{r_{PL}^3} + \frac{m_S}{r_{PS}^3}\right)z. \end{cases} \tag{3.12}$$

I com a hamiltonià,

$$H_{PBC}(x,y,z,p_x,p_y,p_z) = \frac{1}{2}(p_x^2+p_y^2+p_z^2) + yp_x - xp_y - \frac{1-\mu}{r_{PT}} - \frac{\mu}{r_{PL}} - \frac{m_S}{r_{PS}} - \frac{m_S}{a_S^2}(y\sin\theta - \cos\theta). \quad (3.13)$$

Demostració.

Les equacions surten directes de (3.7) utilitzant un nou canvi de variable $\theta := t(1-n_S) = w_S t$ i definint els moments com $p_x = \dot{x} - y$, $p_y = \dot{y} + x$, $p_z = \dot{z}$. □

Observació 3.10. Considerant que el hamiltonià del problema restringit de tres cossos és,

$$H_{PRTC}(x,y,z,p_x,p_y,p_z) = \frac{1}{2}(p_x^2+p_y^2+p_z^2) + yp_x - xp_y - \frac{1-\mu}{r_{PT}} - \frac{\mu}{r_{PL}}. \quad (3.14)$$

Podem reescriure el hamiltonià del problema bicircular com una pertorbació d'aquest,

$$H_{PBC} = H_{PRTC} - \frac{m_S}{r_{PS}} - \frac{m_S}{a_S^2}(y\sin\theta - \cos\theta).$$

4 Mètodes numèrics

En aquest capítol ens centrarem a definir tots els mètodes numèrics i algoritmes que utilitzarem per a trobar òrbites periòdiques en el model bicircular. L'aplicació pròpia d'aquests algoritmes es pot trobar en els annexos **B**, junt amb una breu explicació que fa cada codi.

4.1 Mètodes Runge-Kutta

Com que no podem obtenir una solució analítica per a les equacions de moviment del model de tres cossos, ni tampoc, les del model bicircular, ens caldrà un integrador numèric per a obtenir una aproximació prou bona del flux d'aquestes equacions diferencials. El mètode Runge-Kutta serà el nostre integrador numèric que ens permetrà obtenir una aproximació de les solucions d'una equació diferencial qualsevol i, en particular, la del model bicircular.

4.1.1 Plantejament de l'integrador numèric

Donat un punt inicial conegut x_0 i el problema de valor inicial:

$$\begin{cases} x'(t) = f(t, x(t)), \\ x(t_0) = x_0. \end{cases}$$

Suposem que tenim definida l'equació diferencial associada al problema de valor inicial anterior en l'interval de temps $[a, b]$, amb $t_0 = a$, i volem obtenir un dibuix aproximat de la seva òrbita. Llavors, podem subdividir convenientment aquest interval en petits trossets de longitud h i, utilitzant que coneixem x_0, anar aproximant els següents valors de $x(t)$.

Definició 4.1. *Anomenen* ***mètode d'integració d'un pas*** *als mètodes que depenen del pas anterior per a realitzar una aproximació de la solució. És a dir, per a realitzar l'aproximació x_{n+1} ens cal tenir l'aproximació x_n $\forall n \geq 0$.*

Una forma simple de realitzar aquestes aproximacions és amb el següent mètode:

Definició 4.2. *El* ***mètode d'Euler*** *és un mètode d'integració numèrica d'un pas que aproxima un punt x_{n+1} de l'òrbita mitjançant la tangent de l'òrbita en x_n, fent-la avançar un pas $h > 0$ petit.*

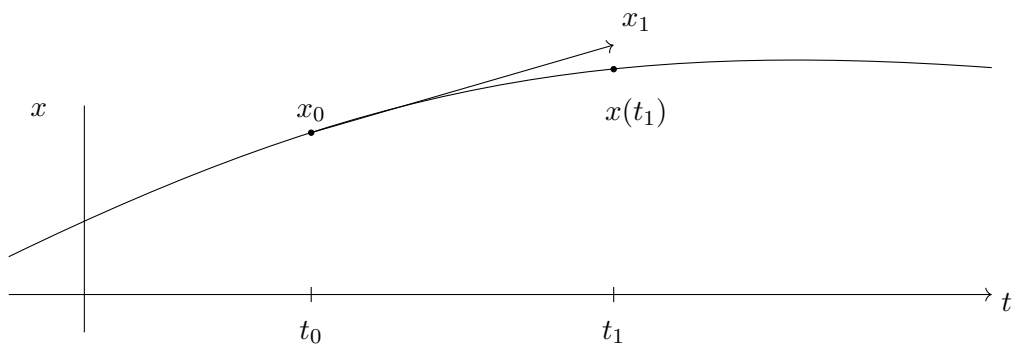


Figura 10: Esquema d'un pas del mètode d'Euler.

Observació 4.3. Desenvolupant per Taylor podem veure que el mètode d'Euler té ordre 1.

De fet, utilitzant el desenvolupament de Taylor podem obtenir mètodes d'ordre el que desitgem. Aquests mètodes, anomenats *mètodes de Taylor*, no els tractarem aquí, però és pot trobar informació a [9].

En general, com més gran sigui l'ordre millor precisió obtindrem. De forma equivalent, com més gran sigui l'ordre més complicat serà el mètode de programar, ja que haurem de calcular les diferencials del camp fins a l'ordre que desitgem. El principal avantatge del mètode Runge-Kutta és que ens estalviarem tots aquests càlculs de derivades, de fet, en canviar el model només caldrà modificar la funció que avalua al camp.

Definició 4.4. *Els algoritmes **Runge-Kutta** són de la forma:*

$$x_0 = x(a),$$

$$x_{n+1} = x_n + h\sum_{i=0}^{k} c_i k_i^n,$$

on $k \in N$ està prefixada a l'hora de buscar el mètode. Les $c_i, i = 1, \ldots, k$ són constants a determinar i les $k_i^n = k_i^n(t_n, x_n, h)$, $i = 1, \ldots, k$ són funcions definides recurrentment per:

$$k_1^n = f(t_n, x_n),$$

$$k_i^n = f(t_n + a_i h, x_n + h\sum_{j=1}^{i-1} b_{ij} k_j^n), i = 2...k,$$

amb a_i, b_{ij} constants a determinar, de manera que el mètode d'integració tingui ordre màxim.

Cal anotar que k serà el nombre d'avaluacions del camp a cada pas, això és degut al fet que el mètode és d'un pas i, per tant, per cada càlcul de x_{n+1} haurem de trobar unes noves k_i^n.

4.1.2 Runge-Kutta d'ordre 2

Presentarem ara una versió de menor ordre (en conseqüència, precisió) de Runge-Kutta amb l'objectiu de mostrar un exemple il·lustratiu. Busquem, doncs, el mètode Runge-Kutta d'ordre màxim que podem obtenir mitjançant dues avaluacions del camp de l'equació diferencial. Tenim doncs,

$$\phi(t, x, h) = c_1 k_1 + c_2 k_2 = c_1 f(t, x) + c_2(f(t + a_2 h, x + h b_{21} f(t, x)).$$

Aplicant el desenvolupament per sèries de Taylor:

$$\phi(t, x, h) = c_1 f(t, x) + c_2(f(t, x) + a_2 h f_t(t, x) + h b_{21} f_x(t, x) f(t, x) + \frac{(a_2 h)^2}{2} f_{tt}$$

$$+ a_2 b_{21} h^2 f_{tx}(t, x) f(t, x) + \frac{(h b_{21})^2}{2} f_{xx}(t, x) f^2(t, x) + O(h^3).$$

Novament, desenvolupem per Taylor $x(t+h)$ al voltant de t i calculem $\frac{x(t+h)+x(t)}{h}$,

$$\frac{x(t+h)+x(t)}{h} = x'(t) + \frac{h}{2!}x''(t) + \frac{h^2}{3!}x'''(t) + O(h^3).$$

Les $x''(t)$ i $x'''(t)$ podem escriure-les amb la igualtat $x'(t) = f(t+h)$ del problema de valor inicial i utilitzant la sèrie de Taylor:

$$\frac{x(t+h)+x(t)}{h} = f(t,x) + \frac{h}{2}(f_t(t,x) + f_x(t,x)f(t,x)) + \frac{h^2}{3!}(f_{tt}(t,x) + 2f_{tx}f(t,x)$$
$$+f_t(t,x)f_x(t,x) + f_x x(t,x)f^2(t,x)) + O(h^3).$$

Sabem que l'ordre d'un integrador és:

$$\frac{x(t+h)+x(t)}{h} - \phi(t,x(t),h) = O(h^p).$$

Si fem el càlcul,

$$(1-c_1-c_2)f(t,x) + \left((\frac{1}{2}-a_2c_2)f_t(t,x) + (\frac{1}{2}-c_2b_{21})f(t,x)f_x(t,x)\right)h +$$
$$+\left((\frac{1}{6}-\frac{a_2^2c_2}{2})f_{tt}(t,x) + (\frac{1}{6}-\frac{b_{21}^2c_1}{2})f_{xx}(t,x)f^2(t,x) + \frac{1}{6}(f_t(t,x)f_x(t,x) + f_x^2(t,x)f(t,x))\right)h^2$$
$$+O(h^3).$$

Com que el terme $(f_t f_x + f_x^2 f)\frac{h^2}{6}$ no serà generalment nul, l'ordre màxim que podrem assolir serà 2. El sistema de constants associat a buscar aquest ordre s'obté d'igualar les constants a 0 per tal d'anul·lar tots els termes d'ordre 0 i 1 (respecte h).

$$\begin{cases} 1-c_1-c_2 = 0, \\ \frac{1}{2}-a_2c_2 = 0, \\ \frac{1}{2}-c_2b_{21} = 0. \end{cases}$$

Un sistema lineal de 3 equacions amb 4 incògnites té infinites solucions. Així doncs, sigui quina sigui l'elecció de les nostres constants, sempre que compleixin el sistema anterior tindrem un Runge-Kutta d'ordre 2.

4.1.3 Control de pas. RK45F

Una de les eleccions més importants, si no la que més, és l'elecció de la h. Definit un error $e > 0$ és clar que sempre que les derivades siguin finites $\exists h > 0$ prou petita tal que l'error obtingut pel mètode sigui més petit que la nostra elecció e. Ara bé, escollir la mateixa h per totes les iteracions no és una solució eficient. La trajectòria pot variar molt o poc depenen del cas que ens trobem. Si escollim una h prou petita tal que l'error resultant sigui el desitjat al llarg de tota la trajectòria, estaríem realitzant moltíssimes iteracions extres que afectaríem l'eficiència del nostre programa.

La solució al nostre problema d'eficiència no serà més que realitzar un càlcul de la h per cada iteració del nostre algoritme. Com que el mètode Runge-Kutta és d'un pas, no tindrem cap problema amb canviar la h. L'elecció, però, no serà trivial. Sigui doncs

una successió $t_0, t_1, t_2 ... t_{n+1}$ no equiespaiada, denotarem h_n per la n-èssima partició on $h_n = t_{n+1} - t_n$.

El mètode *Runge-Kutta-Fehlberg* utilitza dos mètodes Runge-Kutta (un d'un ordre superior al l'altre) i compara els resultats. Donat un punt $x(t_n)$ obtenim dues aproximacions, si aquestes no són prou semblants, reduïm el pas h_n. Per tant, controlarem el valor de la h_n per cada pas.

Que les aproximacions coincideixin prou bé ho fixa l'usuari i ho mesurem amb la diferència de les aproximacions. Les avaluacions del camp del Runge-Kutta de major ordre s'aprofitaran per al de menor ordre. Així doncs, a efectes pràctics, estarem pagant el cost del mètode Runge-Kutta d'ordre més gran.

Ara sí, presentarem el mètode **Runge-Kutta-Fehlberg d'ordre 4 i 5**, que a partir d'ara anomenarem per **RK45F**. Aquest mètode aprofita 6 avaluacions del camp per fer un RK5 i RK4 per estimar la h_n. Hem valorat que l'integrador és prou eficient pels nostres interessos al llarg d'aquest treball.

Donat un problema de valor inicial,

$$\begin{cases} x'(t) = f(t, x(t)), \\ x(t_0) = x_0. \end{cases}$$

L'elecció de les k són:

$$\begin{aligned}
k_1 &= h_n f(t_n, x_n), \\
k_2 &= h_n f(t_n + \frac{1}{4}h_n, x_n + \frac{1}{4}k_1), \\
k_3 &= h_n f(t_n + \frac{3}{8}h_n, x_n + \frac{3}{32}k_1 + \frac{9}{32}k_2), \\
k_4 &= h_n f(t_n + \frac{12}{13}h_n, x_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3), \\
k_5 &= h_n f(t_n + h_n, x_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4), \\
k_6 &= h_n f(t_n + \frac{1}{2}h_n, x_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{50}k_5),
\end{aligned}$$

D'on obtenim el Runge-Kutta 4:

$$\overline{x}_{n+1} = x_n + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5.$$

I el Runge-Kutta 5:

$$\hat{x}_{n+1} = x_n + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6.$$

L'error be definit:

$$\|\hat{x}_{n+1} - \overline{x}_{n+1}\| = \left\| \frac{1}{360}k_1 - \frac{128}{4275}k_3 - \frac{2197}{7240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6 \right\|.$$

Control de pas:

$$|h_{n+1}| = c|h_n| \sqrt[5]{\frac{\epsilon}{\|\overline{x}_{n+1} - \hat{x}_{n+1}\|}}.$$

On ϵ és el valor de precisió desitjat i c és un valor de seguretat que situarem per 0.9.

4.2 Algoritme d'òrbites periòdiques

Per a trobar òrbites periòdiques suposarem que ens situem en un punt prou proper d'una. En aquesta secció definirem l'algoritme implementat per tal d'obtenir els punts de l'òrbita periòdica de període τ començant per un punt p donat. Abans, però, començarem recordant un parell de definicions bàsiques.

Definició 4.5. *Sigui f una funció Lipschitz respecte x definida a $\mathbb{R}\times U$ on U és un obert,*

$$\dot{x} = f(t, x(t)), \tag{4.1}$$

*i sigui $\phi(t,x)$ la solució única de (4.1). Llavors, per $x \in U$, anomenem a la corba $x(t) = \phi(t,x)$, $\forall t \in I \subseteq U$ com a l'**òrbita** de x.*

Definició 4.6. *Sigui $x(t)$ una òrbita, diem que és **periòdica** si existeix almenys un valor $\tau \in \mathbb{R}^+$ tal que $\phi(t,x) = \phi(t+\tau, x)$. Li diem **període** de l'òrbita al valor τ_0 més petit tal que compleix la propietat anterior.*

Ara que ja hem recordat les definicions de l'objecte que estem cercant, definirem l'algoritme per trobar òrbites periòdiques.

Considerem una equació diferencial del tipus (4.1) i volem trobar òrbites periòdiques de τ a prop d'un punt p donat.

Observació 4.7. Escollirem el punt p en tots els casos com un punt d'equilibri del problema de tres cossos restringit.

Com ja sabem, el model bicircular pertorba el model de tres cossos de forma no trivial i, de fet els punts de Lagrange no es troben en el model bicircular, però, en canvi, hi existeixen òrbites periòdiques a prop d'ells.

Per altra part, escollirem el període $\tau = \frac{2\pi}{\omega_S}$ ja que ens apareix implícitament a les equacions del model bicircular.

(i) Donada una condició inicial $x(0) = (x_1(0), x_2(0), ..., x_n(0))$ considerarem la secció de Poincaré temporal:

$$\begin{gathered} P : \mathbb{R}^n \longmapsto \mathbb{R}^n, \\ (x_1(0), x_2(0), ..., x_n(0)) \longmapsto (x_1(\tau), x_2(\tau), ..., x_n(\tau)). \end{gathered} \tag{4.2}$$

Aquesta aplicació P dependrà de l'equació diferencial que estiguem tractant, ja sigui l'associada amb les equacions de moviment del problema de tres cossos o bé del problema bicircular. Ambdues no ens permeten escriure explícitament l'aplicació de Poincaré, per això necessitem utilitzar un integrador numèric. En el nostre cas farem ús del Runge-Kutta presentat a la secció anterior.

Relacionarem els dos models mitjançant les equacions (3.7) del problema bicircular i un paràmetre $\epsilon \in [0,1]$ que multipliqui els termes de l'equació que continguin la massa del Sol.

Definició 4.8. *Les equacions que utilitzarem per a l'integrador Runge-Kutta són:*

$$\begin{cases} \dot{x} = \ddot{x}, \\ \dot{y} = \ddot{y}, \\ \ddot{x} = 2\dot{y} + x - \frac{1-\mu}{r_{PT}^3}(x-\mu) - \frac{\mu}{r_{PL}^3}(x-\mu+1) - \frac{\epsilon m_S}{r_{PS}^3}(x - a_S\cos(t-n_S t)) - \frac{\epsilon m_S}{a_S^2}\cos(t-n_S t), \\ \ddot{y} = -2\dot{x} + y - \frac{1-\mu}{r_{PT}^3}y - \frac{\mu}{r_{PL}^3}y - \frac{\epsilon m_S}{r_{PS}^3}(y + a_S\sin(t-n_S t)) + \frac{\epsilon m_S}{a_S^2}\sin(t-n_S t). \end{cases} \tag{4.3}$$

Anomenarem f_1 i f_2 a les equacions corresponents a $\ddot{x}$ i $\ddot{y}$ respectivament.

Observació 4.9. Negligim la coordenada z perquè estem tractant el moviment de tres cossos restringit en el pla.

(ii) Considerarem ara un punt fix de l'aplicació P. En efecte, si x' és un punt fix de P, $P(x') = x'$ llavors l'òrbita de x' serà periòdica amb període τ.

Observació 4.10. Hem reduït el problema d'obtenir òrbites periòdiques a trobar zeros de l'aplicació $F(z) := P(z) - z$, on P és l'aplicació de Poincaré associada l'EDO.

(iii) Per trobar zeros de la funció F utilitzarem el mètode de Newton. Com estem treballant en dimensions superiors a 1 tindrem que les diferencials dels sistemes no seran trivials i haurem de fer ús de l'integrador RK45F. Recordem que el mètode de Newton en diverses variables és,

$$x_{k+1} = x_k + z, \tag{4.4}$$

On z és la solució del sistema lineal $DF(z)z = -F(z)$. La diferencial de la funció F és $DP - Id$ on Id és la identitat. Per trobar DP necessitarem les equacions variacionals respecte a la condició inicial z_0.

Considerant el següent problema de Cauchy:

$$\begin{cases} z'(t) = f(t, z(t)), \\ z(a) = z_0. \end{cases}$$

Derivant respecte a les condicions inicials es té:

$$\begin{cases} \frac{\partial}{\partial t}\left[D_{z_0} z(t, z_0)\right] = D_z f(t, z(t, z_0))\left[D_{z_0} z(t, z_0)\right], \\ D_{z_0} z(t_0, z_0) = Id. \end{cases} \tag{4.5}$$

Utilitzarem el RK45F per obtenir resoldre el sistema numèricament. Reescriurem (4.5) com:

$$\begin{cases} \dot{V} = D_z f(t, z) V, \\ V(t_0) = Id. \end{cases} \tag{4.6}$$

Definició 4.11. *L'algoritme descrit pels passos **i), ii)** i **iii)** consisteix en el nostre **algoritme de cerca d'òrbites periòdiques**.*

Observació 4.12. La diferencial $D_z f(t, z)$ és coneguda.

$$D_z f(t, z) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{df_1}{dx} & \frac{df_1}{dy} & 0 & 2 \\ \frac{df_2}{dx} & \frac{df_2}{dy} & -2 & 0 \end{pmatrix}. \tag{4.7}$$

Teorema 4.13. ***Fórmula de Liouville**. Sigui $\phi(t)$ una matriu quadrada de dimensió $n \times n$ tal que verifica la següent equació homogènia de primer ordre $\phi'(t) = A(t)\phi(t)$, $t \in I$ on I és un interval de $\mathbb{R}$. Llavors, si la traça de la matriu $A(t)$ és contínua en I, es compleix que:*

$$\det \phi(t) = \det \phi(t_0) \exp\left(\int_{t_0}^{t} \operatorname{Tr} A(s)\, ds \right). \tag{4.8}$$

Amb $t, t_0 \in I$.

Demostració.

Podem calcular la derivada del determinant de $\phi = (\phi_{i,j})_{i,j\in\{0,\dots,n\}}$ fent ús de la fórmula de Leibniz,

$$(\det \phi(t))' = \sum_{i=1}^{n} \det \begin{pmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \vdots & \vdots & & \vdots \\ \phi'_{i1} & \phi'_{i2} & \dots & \phi'_{1n} \\ \vdots & \vdots & & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} \end{pmatrix}. \tag{4.9}$$

Sabem que $\phi(t)$ compleix l'equació $\phi'(t) = A(t)\phi(t)$, és a dir,

$$\phi'_{ik} = \sum_{j=1}^{n} a_{ij}\phi_{jk}.$$

Mirant per files tenim

$$(\phi'_{i1}, \phi'_{i2}, \dots, \phi'_{in}) = \sum_{j=1}^{n} a_{ij}(\phi_{j1}, \phi_{j2}, \dots, \phi_{jn}), \quad i \in \{1, \dots, n\}.$$

Sabem que si a una fila d'una matriu li apliquem una combinació lineal d'altres files, el determinant no varia. Per tant,

$$\det \begin{pmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \vdots & \vdots & & \vdots \\ \phi'_{i1} & \phi'_{i2} & \dots & \phi'_{1n} \\ \vdots & \vdots & & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} \end{pmatrix} = \det \begin{pmatrix} \phi_{11} & \dots & \phi_{1n} \\ \vdots & & \vdots \\ \phi'_{i1} - \sum_{j=1, j\neq i}^{n} a_{ij}\phi_{j1} & \dots & \phi'_{1n} - \sum_{j=1, j\neq i}^{n} a_{ij}\phi_{jn} \\ \vdots & & \vdots \\ \phi_{n1} & \dots & \phi_{nn} \end{pmatrix}$$

$$= \det \begin{pmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \vdots & \vdots & & \vdots \\ a_{ii}\phi'_{i1} & a_{ii}\phi'_{i2} & \dots & a_{ii}\phi'_{1n} \\ \vdots & \vdots & & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} \end{pmatrix} = a_{ii} \det \phi.$$

Fent ús de (4.9), obtenim:

$$(\det \phi(t))' = \sum_{i=1}^{n} a_{ii} \det \phi = \operatorname{Tr} A \det \phi. \tag{4.10}$$

Resta veure que (4.10) implica la fórmula de Liouville. Farem servir com a notació $u(t) = \det \phi(t)$ i $\alpha(t) = \operatorname{Tr} A$, llavors,

$$u'(t) = u(t)\alpha(t) \leftrightarrow \frac{du}{dt} = u(t)\alpha(t).$$

Resolent per variables separables,

$$\frac{du}{u(t)} = \alpha(t)dt \to \int \frac{du}{u(t)} = \int \alpha(t)dt.$$

$$\log u(t) = \int \alpha(t)dt + C.$$

Posant $t = t_0$,

$$\log u(t_0) = \int_{t_0}^{t_0} \alpha(t)dt + C = C.$$

En definitiva,

$$\log u(t) = \int_{t_0}^{t} \alpha(t)dt + \log u(t_0).$$

Que implica,

$$\frac{u(t)}{u(t_0)} = \exp\Big(\int_{t_0}^{t} \alpha(s)\, ds \Big).$$

Essent la fórmula de Liouville. □

Observació 4.14. Una forma numèrica de comprovar que el sistema (4.6) està ben programat és calcular el determinant de la matriu variacional quan el temps és igual al període $T = \frac{2\pi}{\omega_s}$. Ja que, per definició del sistema, i, aplicant la fórmula de Liouville, sabem que aquesta haurà de donar exactament 1.

4.3 Mètode de continuació

Veurem en el pròxim capítol com el fet de jugar amb la pertorbació generada pel Sol ens permet trobar diferents òrbites periòdiques en un mateix punt d'equilibri. Per tal d'experimentar amb aquesta pertorbació hem de moure el valor de la variable ϵ i, en conseqüència, definir un mètode de continuació. Precisament, en aquesta secció presentarem el mètode de continuació emprat al programari.

Considerem la corba $F(\hat{x}, \epsilon) = 0$ on $\hat{x} = (x, y, \dot{x}, \dot{y}) \in \mathbb{R}^4$ i $\epsilon \in [0, 1]$ de la qual volem trobar-hi més punts. Denotem $y := (\hat{x}, \epsilon)$ i suposem que coneixem dos punts de la corba y_{-1} i y_0. Podem definir el següent vector:

$$v = \frac{(y_0 - y_{-1})}{||y_0 - y_{-1}||_2}.$$

Realitzarem un pas de predicció al següent punt de la corba mitjançant aquest vector. Sigui $\delta > 0$ un valor prou petit, definim les noves prediccions de punts com:

$$\hat{y}_k = \hat{y}_{k-1} + \delta v.$$

Observació 4.15. En general, el punt $\hat{y}_k$ no serà un punt de la corba que busquem, però si prou proper perquè el mètode corrector que apliquem convergeixi.

En el nostre cas utilitzarem com a mètode corrector el mètode de Newton amb el següent sistema,

$$\begin{cases} F(y) = 0, \\ ||y - y_{k-1}||_2^2 - \delta^2 = 0. \end{cases}$$

Els punts inicials els obtindrem aplicant l'algoritme de cerca amb els paràmetres ϵ fixes que desitgem. Si volem moure ϵ de 0 a 1 haurem d'escollir $y_{-1} = (\hat{x}, 0)$ i $y_0 = (\hat{x}, \delta)$ per tal de definir la direcció del vector tangent de forma que el valor d'èpsilon augmenti. Altrament, escolliríem $y_{-1} = (\hat{x}, 1)$ i $y_0 = (\hat{x}, 1 - \delta)$.

Observació 4.16. Sigui la iteració k-èssima, les derivades de l'equació de l'esfera segons els paràmetres x, y, $\dot{x}$, $\dot{y}$ i ϵ són:

$$2(x - x_{k-1}), \quad 2(y - y_{k-1}),$$

$$2(\dot{x} - \dot{x}_{k-1}), \quad 2(\dot{y} - \dot{y}_{k-1}), \quad 2(\epsilon - \epsilon_{k-1}).$$

Observació 4.17. Les derivades de $F(\hat{x}, e)$ respecte d'èpsilon s'obtenen a partir de les variacionals respecte del paràmetre ϵ. Si tornem a fixar-nos en les equacions (4.3) veurem que podem escriure $F(\hat{x}, e) = f(\hat{x}) + g(\hat{x})$ de tal manera que,

$$\frac{dF}{d\epsilon} = D_{\hat{x}}F(\hat{x}, \epsilon) + \frac{\partial F}{\partial \epsilon}.$$

On $D_{\hat{x}}F(\hat{x}, \epsilon)$ és la diferencial vista a (4.7) i $\frac{\partial F}{\partial \epsilon} = g(\hat{x})$.

4.4 Mètode del Tir Múltiple

A la secció **2.4.2** vàrem veure que els punts col·lineals que eren linealment inestables. Aquesta inestabilitat afectarà a l'integrador numèric de manera que l'error al calcular $P(z)$ creixerà molt ràpidament i el mètode de Newton plantejat en el programari no ens convergirà. Per tant, ens caldrà plantejar un mètode encara més refinat per tal d'aconseguir que el mètode de Newton convergeixi. Presentarem, doncs, el mètode de Tir Múltiple.

Sigui $I := [a, b]$ un interval real, i sigui:

$$\begin{cases} y' = f(t, y), & \forall t \in I, \\ y(t_0) = s_0, \end{cases} \tag{4.11}$$

un problema de valor inicial amb solució $y(t)$. Considerem una partició equiespaiada de l'interval I en n trossos: $P(I) := \{t_0, t_1 \ldots, t_n\}$ tal que $a = t_0 < t_1 < \ldots < t_n = b$. Suposem que $y(t_k) = s_k$ per a tot $k \in \{0, 1 \ldots n\}$ per uns certs s_k desconeguts.

Denotant $y(t; t_k, s_k)$ com la solució de:

$$\begin{cases} y' = f(t, y), & t \in [t_k, t_{k+1}], \\ y(t_k) = s_k, \end{cases}$$

i imposant que:

$$\begin{cases} y(t; t_k, s_k) = s_{k+1}, & \forall k \in \{0, \ldots, n-1\}, \\ y(t; t_{n-1}, s_{n-1}) = s_0. \end{cases}$$

podem definir la funció contínua F formada pels trossos $y(t; t_k, s_k)$ (superposats). Aquesta funció és la solució del problema de valor inicial (4.11). Fixem-nos que ara podem reescriure el problema de resoldre (4.11) com el problema de trobar els vectors s_k. Per tant, hem de resoldre simultàniament els sistemes:

$$F(t) := \begin{bmatrix} F_0(s_0, s_1) \\ F_1(s_1, s_2) \\ F_2(s_2, s_3) \\ \vdots \\ F_{n-1}(s_{n-1}, s_n) \\ F_n(s_n, s_0) \end{bmatrix} := \begin{bmatrix} y(t; t_0, s_0) - s_1 \\ y(t; t_1, s_1) - s_2 \\ y(t; t_2, s_2) - s_3 \\ \vdots \\ y(t; t_{n-1}, s_{n-1}) - s_n \\ y(t; t_n, s_n) - s_0 \end{bmatrix} = 0. \tag{4.12}$$

Definició 4.18. *Anomenarem* ***Tir Múltiple*** *al mètode que resol (4.11) mitjançant el càlcul de l'arrel de (4.12).*

El mètode del Tir múltiple transforma el problema de valor inicial en la solució simultània de $n+1$ problemes de trobar una arrel, el qual el podem resoldre mitjançant el mètode de Newton; seguint la definició (4.18) i sigui $s = (s_0, s_1, \ldots, s_n)$, amb una aproximació del vector $s \approx s^0$ podem iterar pel mètode de Newton tal que:

$$s^{(i+1)} = s^{(i)} - \left[DF(s^{(i)})\right]^{-1} F(s^{(i)}), \quad i = 0, 1, \ldots$$

Observació 4.19. En el nostre cas els punts s_k són tots iguals al punt d'equilibri que considerem, ja que, per definició, qualsevol punt d'equilibri és constant al llarg del temps respecte al seu sistema d'equacions diferencials. D'aquí obtenim l'aproximació inicial essent totes les $s_k = L_1$.

Observació 4.20. Tot i que hem presentat el mètode del Tir Múltiple per n particions arbitraries, només ens en caldran 4 perquè l'error derivat del Runge-Kutta no sigui massa gran.

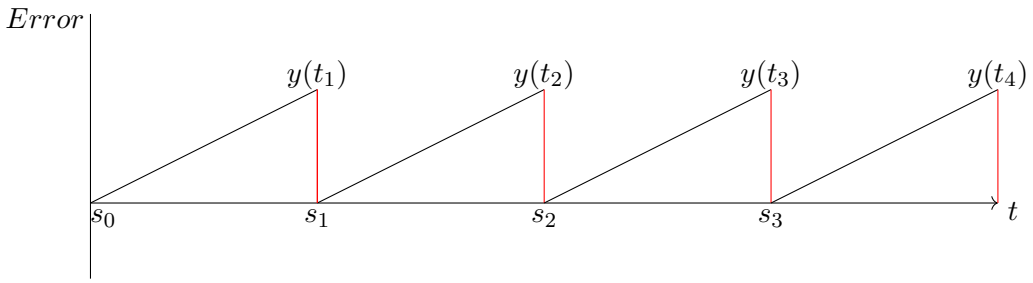


Figura 11: Esquema del creixement de l'error de l'integrador respecte el temps.

L'algoritme d'òrbites periòdiques variarà lleugerament. L'aplicació descrita a (ii) $F(z) := P(z) - z$ d'on apliquem el mètode de Newton, passarà a quatre aplicacions:

$$\begin{cases} F_0(x) := P(x) - y, & t \in [0, \frac{\tau}{4}], \\ F_1(y) := P(y) - z, & t \in [\frac{\tau}{4}, [\frac{2\tau}{4}], \\ F_2(z) := P(z) - s, & t \in [\frac{2\tau}{4}, [\frac{3\tau}{4}], \\ F_3(s) := P(s) - x, & t \in [\frac{3\tau}{4}, \tau], \end{cases}$$

On $x := s_0$, $y := s_1$, $t := s_2$, $s := s_3$.

La matriu diferencial que obtindrem pel mètode de Newton en aquest cas serà una matriu 16×16 on tindrem caixes de 4×4 corresponents a les diferencials de P, la identitat o bé directament la matriu nul·la. En definitiva, la diferencial D tindrà la següent forma:

$$D = \begin{pmatrix} D_x F_0 & -I_d & 0 & 0 \\ 0 & D_y F_1 & -I_d & 0 \\ 0 & 0 & D_z F_2 & -I_d \\ -I_d & 0 & 0 & D_s F_3 \end{pmatrix}.$$

5 Òrbites periòdiques

En aquest capítol presentarem diverses òrbites periòdiques del model bicircular en el sistema Terra-Lluna-Sol trobades amb el programari de l'apartat **B** dels annexos.

Al capítol **3** hem demostrat que es pot tractar el model bicircular com una pertorbació del model restringit de tres cossos mitjançant les equacions de moviment. De fet, les òrbites periòdiques que presentarem es troben al voltant dels punts L_1 i L_4.

5.1 Òrbites periòdiques al voltant de L_4

Per la secció **2.4** ja sabem que els únics punts d'equilibri linealment estables són L_4 i L_5 quan la massa μ del sistema és menor que la massa de Routh μ_0. En el cas del sistema Terra-Lluna $\mu = 0.012505819187 < \mu_0$ i, per tant, es compleix que els punts equilaterals són estables. L'estabilitat ens permetrà fer ús del Runge-Kutta sense experimentar errors grans al llarg de la integració i procedir amb l'estratègia general explicada a l'anterior secció.

Observació 5.1. L'òrbita que obtindrem a partir del punt L_4 presenta una simetria respecte de L_5. Això és conseqüència directa de les equacions del problema de tres cossos i del fel que els punts siguin simètrics respecte a l'eix de les x's.

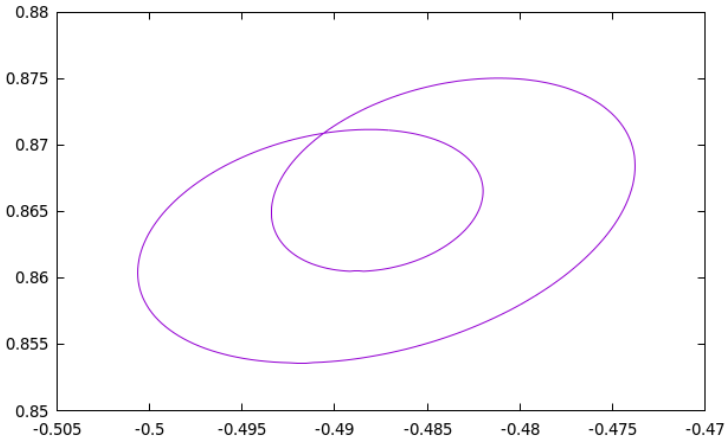


Figura 12: Òrbita periòdica al voltant de L_4.

El codi emprat per a trobar l'òrbita periòdica també ens troba, mitjançant un programari extern, els seus valors propis. En concret, per l'òrbita descrita a la figura 12 obtenim els següents:

$$-0.48162 + 0.87638i, \quad -0.48162 - 0.87638i,$$
$$+0.89431 + 0.0000i, \quad +1.11816 + 0.0000i,$$

on els darrers dos vaps impliquen la inestabilitat lineal de la mateixa òrbita periòdica.

Aplicant el mètode de continuació movent e de 0 fins a 1 obtindrem una nova òrbita periòdica. De la mateixa manera, començant per $e = 1$ obtenim la tercera òrbita periòdica.

En el cas de l'òrbita il·lustrada a la figura 13 obtenim els següents valors propis:

$$-0.66042 + 0.75089i \quad -0.66042 - 0.75089i$$

$$0.98690 + 0.16131i \quad 0.98690 - 0.16131i$$

En aquest cas podem afirmar que és estable linealment.

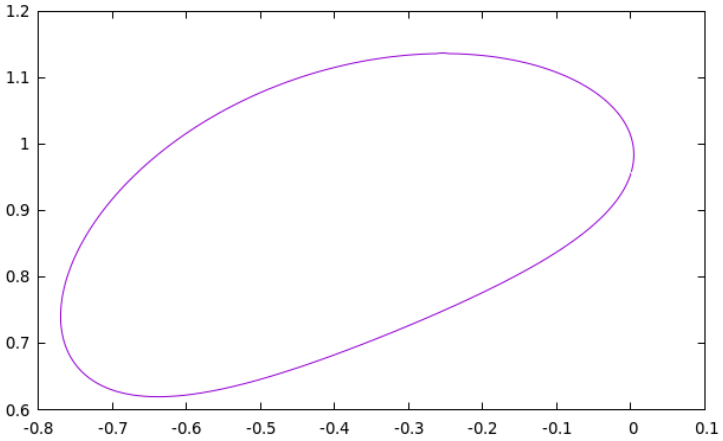


Figura 13: Segona òrbita periòdica al voltant de L_4.

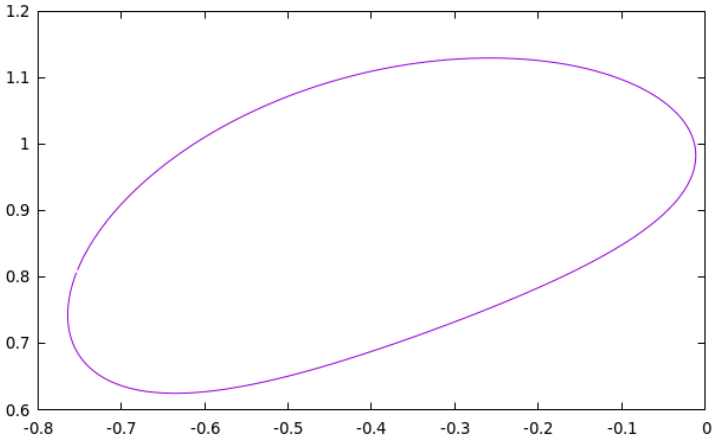


Figura 14: Tercera òrbita periòdica al voltant de L_4.

La tercera òrbita periòdica també és linealment estable, ja que els seus valors propis són:

$$-0.65254 + 0.75775i \quad -0.65254 - 0.75775$$

$$0.98766 + 0.15659i \quad 0.98766 - 0.15659i$$

Un fenomen també curiós és el descrit a la figura 15, quan comencem pel valor $\epsilon = 1$ i el volem fer decréixer aquest es troba amb un punt de retorn.

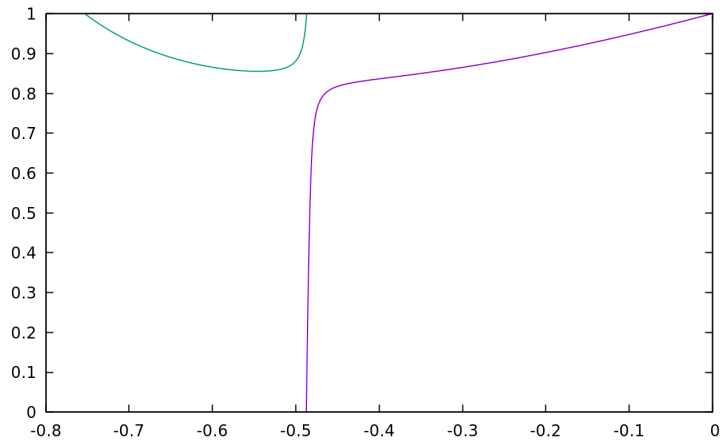


Figura 15: La corba verda fa referència a l'evolució de l'èpsilon segons la coordenada x quan comencem la continuació per $\epsilon = 1$, mentre que la lila quan comencem per $\epsilon = 0$.

5.2 Òrbites periòdiques al voltant de L_1

Com bé ja vàrem comentar, en el cas dels punts col·lineals cal refinar el mètode de l'integrador utilitzant el mètode del Tir múltiple. En concret, tallem la secció en 4 trossos de temps $\frac{1}{4}\tau$, on τ és el període. L'òrbita obtinguda és la següent:

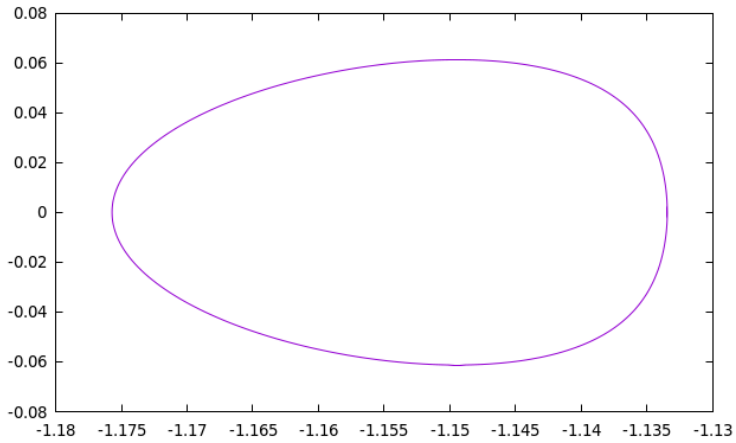


Figura 16: Òrbita periòdica al voltant de L_1.

En aquest cas el paràmetre es troba amb un punt de retorn quan $e \approx 0.55$.

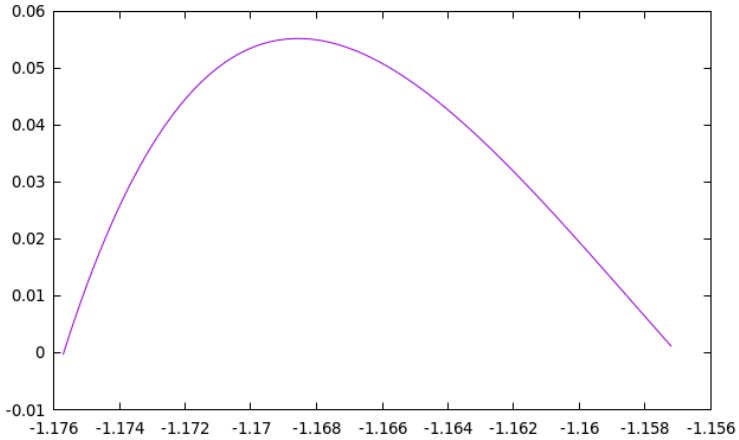


Figura 17: Evolució del paràmetre èpsilon segons la coordenada x, pel punt L_1.

6 Conclusions

L'objectiu principal d'aquest treball consistia a estudiar el problema de tres cossos restringit, el model bicircular, i, en última instància, cercar diverses òrbites periòdiques en el darrer model.

Iniciàvem el treball amb el problema de tres cossos restringit en el capítol 2. Primerament, obteníem l'equació de moviment en coordenades sinòdiques, d'on trobaríem la constant de Jacobi. Abans d'abordar les conseqüències d'aquesta constant, ens centraríem a demostrar l'existència dels punts de Lagrange, cinc punts d'equilibri que es troben en el problema de tres cossos restringit. Tot seguit, provaríem que només dos d'ells, L_4 i L_5, en són linealment estables quan la $\mu < \mu_0$, essent μ_0 la massa de Routh.

Finalment, tancaríem el capítol 2 amb un estudi de la principal conseqüència de la constant de Jacobi C, les corbes de velocitat zero. Aquestes ens separen les diferents regions de moviment possibles segons el valor de la constant C, ja que per definició divideixen l'espai segons si el mòdul de la velocitat és positiu, o bé, negatiu. Vàrem veure que la forma d'aquestes corbes variava dràsticament depenent del valor de la constant de Jacobi, i, per això mateix, vam realitzar una classificació segons els diversos rangs de valors de C, delimitats pels valors de la constant en els punts d'equilibri.

Per altra banda, en el capítol 3 ens vam centrar en el problema bicircular. A diferència del problema de tres cossos restringit, aquest model s'utilitza per predir el moviment d'una partícula (amb massa negligible) en un sistema que conté tres masses. No obstant això, finalitzàvem el mateix capítol demostrant que el model bicircular és només una pertorbació del model definit pel problema de tres cossos restringit.

Un cop contemplat ambdós models, ens faltava enllestir l'últim objectiu principal del treball: cercar òrbites periòdiques en el model bicircular pel sistema Terra-Lluna-Sol. A fi de dur-la a terme, ens feia falta descriure els diversos mètodes i algoritmes numèrics emprats en el programari per trobar les òrbites periòdiques. Aquest era, doncs, l'objectiu del capítol 4.

En el quart capítol del treball introduiríem els mètodes Runge-Kutta com a integrador numèric, en concret, el RK45F. També definiríem l'algoritme d'òrbites periòdiques i un mètode de continuació per a modificar la influència del Sol. Per acabar, introduíem el mètode de Tir múltiple, que ens serviria per a refinar l'integrador numèric en els casos on es tractava amb punts inestables.

En el darrer capítol presentàvem els diversos resultats de les òrbites periòdiques. Tot comentant les òrbites periòdiques cercades al voltant del punt L_4 i, posteriorment, en el punt L_1.

En conclusió, aquest treball ha resultat molt interessant alhora que exigent a causa de la quantitat de teoria de mètodes numèrics i mecànica celeste que es pot arribar a aplicar. Malgrat les dimensions del treball, s'han assolit els principals objectius, tot i que encara es pot continuar fent un estudi general del problema de tres cossos, com també, una cerca més completa de les òrbites periòdiques en el model bicircular. Ambdues direccions queden obertes al lector interessat.

Referències

[1] Mark, C.: *Energy potential analysis of zero velocity curves in the restricted Three-body Problem.* Thesis, the University of Texas at Austin, 1993.

[2] Dickson, L.: *First course in the theory of equations.* John Wiley & Sons Inc, London, 1922.

[3] González, A.: *Ecuaciones diferenciales I.* `https://teorica.fis.ucm.es/metodos/Apuntes/edi-ag.pdf`, 2004.

[4] Greenspan, T.: *Stability of the Lagrange Points, L_4 and L_5.* `api.semanticscholar.org/CorpusID:45742224`, 2014

[5] Jorba, À.: *The Lagrangian Solutions.* `www.maia.ub.es/dsg/2012/1214jorba.pdf`, 2012.

[6] Koon, W.; Lo, M.; Marsden, J.; Ross, S.: *Dynamical Systems, the Three-Body Problem and Space Mission Design.* Marsden Books, 2022. ISBN 978-0-615-24095-4.

[7] Pollard, H.: *Mathematical Introduction to Celestial Mechanics.* Prentice-Hall Mathematical series, Englewood Cliffs, New Jersey, 1966.

[8] Devaney, R.; Hirsch, M.; Smale, S.: *Differential Equations, Dynamical Systems, and an Introduction to Chaos.* Elsevier Inc, 2013.

[9] Stoer, J.; Bulirsch, J.: *Introduction to Numerical Analysis.* Springer, New York, 1980.

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

A Càlcul dels punts d'equilibri del sistema Terra-Lluna

En aquesta secció es pot trobar càlcul dels punts de Lagrange del cas Terra-Lluna pel problema de tres cossos restringit. Els valors que trobarem són utilitzats per dibuixar l'esquema que es troba a la secció **2.3.3**.

Al llarg d'aquest apartat el valor μ correspondrà al valor de la μ_{TL} Terra-Lluna $\mu_{TL} = 0.012505819187$. Començarem calculant els punts col·lineals, dels quals ja en sabem les equacions de cinquè grau per resoldre associades a cada punt.

Punt L_1

Pel que hem vist al llarg de la secció **2.3.1**, l'equació associada al punt L_1 és:

$$f_\mu(\epsilon) := \epsilon^5 + (3-\mu)\epsilon^4 + (3-2\mu)\epsilon^3 - \mu\epsilon^2 - 2\mu\epsilon - \mu = 0.$$

Amb la nostra μ_{TL}:

$$f_{\mu_{TL}}(\epsilon) := \epsilon^5 + (2.987494180813)\epsilon^4 + (2.974988361626)\epsilon^3 - 0.012505819187\epsilon^2$$

$$-0.025011638374\epsilon - 0.012505819187 = 0.$$

Observació A.1. Per $\epsilon = 0$ obtenim $f_{\mu_{TL}}(0) = -0.012505819187$ i per $\epsilon = 1$ obtenim $f_{\mu_{TL}}(1) > 0$, per tant, pel teorema de Bolzano, tenim que a l'interval $(0,1)$ tenim existeix almenys una arrel. Fet que ja havíem comprovat de forma general.

Per trobar l'arrel farem servir el mètode de Newton iterant des de $\epsilon = 0$ [5]. Obtenim com a resultat $\epsilon = 0.169538634083987$, és a dir, l'abscissa del punt L_1 és:

$$x = \mu - 1 - \epsilon = 0.012505819187 - 1 - 0.169538634083987 = -1.157032814896987.$$

Punt L_2

En el cas del segon punt col·lineal l'equació associada és:

$$f_\mu(\epsilon) := \epsilon^5 - (3-\mu)\epsilon^4 + (3-2\mu)\epsilon^3 - \mu\epsilon^2 + 2\mu\epsilon - \mu = 0.$$

Amb $\mu_{TL} = 0.012505819187$:

$$f_{\mu_{TL}}(\epsilon) := \epsilon^5 - (2.987494180813)\epsilon^4 + (2.974988361626)\epsilon^3 - 0.012505819187\epsilon^2$$

$$+0.025011638374\epsilon - 0.012505819187 = 0.$$

Novament, iterant pel mètode de Newton amb punt inicial $\epsilon = 0$ s'obté una arrel, en aquest cas a $\epsilon = 0.152312968820820$. Finalment l'abscissa que estem buscant és,

$$x = \mu - 1 + \epsilon = 0,012505819187 - 1 + 0,152312968820820 = -0,83518121199218.$$

[5] S'ha realitzat un petit programa que té com a algorítme principal el mètode de Newton en dimensió 1 per aquest apartat.

Punt L_3

L'equació associada al tercer punt col·lineal és:

$$f_\mu(\epsilon) := \epsilon^5 + (2+\mu)\epsilon^4 + (1+2\mu)\epsilon^3 - (1-\mu)\epsilon^2 - 2(1-\mu)\epsilon - (1-\mu) = 0.$$

En el cas Terra-Lluna, l'equació és:

$$f_{\mu_{TL}}(\epsilon) := \epsilon^5 + (2.012505819187)\epsilon^4 + (1.025011638374)\epsilon^3 - (0.987494180813)\epsilon^2$$

$$-(1.974988361626)\epsilon - (0.987494180813) = 0.$$

En aquest cas ens adonem que el punt inicial $\epsilon = 1$ és millor que no pas l'elecció $\epsilon = 0$, per tant, iterarem amb el mètode de Newton amb punt inicial $\epsilon = 1$. En definitiva, l'arrel buscada es troba a $\epsilon = 0.992704831724847$ i, en conseqüència, el punt L_3 té per abscissa:

$$x = \mu + \epsilon = 0.012505819187 + 0.992704831724847 = 1.005210650911847.$$

Punts equilaterals L_4 i L_5

En el cas dels punts equilaterals no ens caldrà fer càlculs numèrics molt complexos. De fet, podem extreure directament les coordenades dels punts utilitzant les posicions de les masses primàries i del fet que L_4 i L_5 formem triangles equilàters ($r_1 = r_2 = 1$).
Utilitzat,

$$\begin{cases} \cos\frac{\pi}{3} = \frac{1}{2}, \\ \sin\frac{\pi}{3} = \frac{\sqrt{3}}{2}, \\ m_1 = (\mu, 0) \quad m_2 = (\mu - 1, 0). \end{cases}$$

Obtenim les posicions dels punts equilaterals:

$$L_4 = (-0.487494180813, \frac{\sqrt{3}}{2}) \quad L_5 = (-0.487494180813, -\frac{\sqrt{3}}{2}).$$

Finalment, recordem que l'esquema es pot trobar a la secció **2.3.3** del treball.

B Implementació del codi emprat al llarg del treball

Al llarg d'aquest apèndix introduirem els diferents codis emprats al llarg del treball. Començarem amb el codi de les corbes de velocitat zero.

Recordem que les corbes de velocitat zero venen definides per:

$$\Omega(x,y) = \frac{C}{2}.$$

Busquem doncs els zeros de la funció:

$$f(x,y) = \left(x^2+y^2\right) + \frac{2\mu}{\sqrt{(x+1-\mu)^2+y^2}} + \frac{2(1-\mu)}{\sqrt{(x-\mu)^2+y^2}} + \mu\,(1-\mu) - C. \quad \text{(B.1)}$$

Ens caldrà doncs definir un mètode de continuació per a buscar zeros de la funció (B.1). Utilitzarem el mètode de continuació **predictor-corrector amb pseudo-paràmetre arc** explicat a l'assignatura de Mètodes Numèrics II.

Per trobar un punt inicial de la corba és realitza un cerca a les semirectes $\{y=0, x>0\}$ amb el mètode de newton en dimensió 1. En cas que la corba de velocitat zero estigui en el rang de mínim valor C cal fer la cerca en les semirectes $\{x=0, y<0\}$ i/o $\{x=0, y>0\}$, ja que la corba no talla l'eix de les x's.

El codi és el següent:

```
/* Autor: Arnau Ruiz Sebastián*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define h 1e-3 /* Distància entre dos punts consecutius de la corba */
#define tol 1e-5 /* Tolerància per les divisions per 0 */
#define mu 0.12141 /*Mu Terra-Lluna*/
#define J 3.22 /*Constant de Jacobi desitjada*/
#define error 1e-12 /*Error pel mètode de Newton*/
#define iterMax 1e5 /*Iteracions màximes pel mètode de Newton*/
#define n 1e6 /*Punts que volem calcular*/

double avaldfx(double x,double y);
double avaldfy(double x,double y);
double avaluaf(double x, double y);
void correccio(double x0, double y0, double *x, double *y);
void metode(double x, double y, FILE* fout);


int main(void){
  int i = 0, trobat = 1;
  double x = 0, y=1, x0, y0, df, f, Nerror = 1;
  FILE *fout;
  fout= fopen("corbes", "w");

  y = 0;
  i = 0;
  //Mètode de Newton a f(x,0), x>0 per trobar el (x0, y0) de la corba de
    velocitat zero inicial
  while (Nerror> error && i < iterMax){
    x0= x;
```

```
33      f = avaluaf(x,0);
34      df = avaldfx(x,0);
35      if(fabs(df) < tol){
36        i = iterMax; //Aturem el mètode de Newton per divisió entre zero
37      }
38      x = x0 - f/df;
39      i++;
40      Nerror = fabs(x - x0);
41
42    }
43    if (i > iterMax){
44      printf("No hem trobat cap punt inicial de la corba de velocitat zero \
        n");
45      trobat = 0;
46    }
47
48
49    /*Mètode Predictor-Corrector*/
50    if(trobat){
51        printf("El punt inicial 1 és (%lf, %lf) \n", x, y);
52        metode(x,y, fout);
53    }
54
55    /*Següent punt*/
56    i = 0;
57    Nerror = 1;
58    y = 0;
59    x = -0.8;
60    trobat = 1;
61
62    while (Nerror> error && i < iterMax){
63      x0= x;
64      f = avaluaf(x,0);
65      df = avaldfx(x,0);
66      if(fabs(df) < tol){
67        i = iterMax; //Aturem el mètode de Newton per divisió entre zero
68      }
69      x = x0 - f/df;
70      i++;
71      Nerror = fabs(x - x0);
72
73    }
74    if (i > iterMax){
75      printf("No hem trobat cap punt inicial de la corba de velocitat zero \
        n");
76      trobat = 0;
77    }
78
79
80    /*Mètode Predictor-Corrector*/
81    if(trobat){
82      printf("El punt inicial 2 és (%lf, %lf) \n", x, y);
83      metode(x,y, fout);
84    }
85
86
87    /*Trobem el següent punt inicial*/
88    i = 0;
89    Nerror = 1;
90    y = 1;
91    x = 0;
```

```
92    trobat = 1;
93
94    while (Nerror > error && i < iterMax){
95      y0= y;
96      f = avaluaf(0,y);
97      df = avaldfy(0,y);
98      if(fabs(df) < tol){
99        i = iterMax; //Aturem el mètode de Newton per divisió entre zero
100     }
101     y = y0 - f/df;
102     i++;
103     Nerror = fabs(y - y0);
104   }
105   if (i > iterMax){
106     printf("No hem trobat cap punt inicial de la corba de velocitat zero \
      n");
107     trobat = 0;
108   }
109
110
111   /*Mètode Predictor-Corrector*/
112   if(trobat){
113     printf("El punt inicial 3 és (%lf, %lf) \n", x, y);
114     metode(x,y, fout);
115   }
116
117   /*Trobem el següent punt inicial*/
118   i = 0;
119   Nerror = 1;
120   y = -1;
121   x = 0;
122   trobat = 1;
123
124   while (Nerror > error && i < iterMax){
125     y0= y;
126     f = avaluaf(0,y);
127     df = avaldfy(0,y);
128     if(fabs(df) < tol){
129       i = iterMax; //Aturem el mètode de Newton per divisió entre zero
130     }
131     y = y0 - f/df;
132     i++;
133     Nerror = fabs(y - y0);
134   }
135   if (i > iterMax){
136     printf("No hem trobat cap punt inicial de la corba de velocitat zero \
      n");
137     trobat = 0;
138   }
139
140
141   /*Mètode Predictor-Corrector*/
142   if(trobat){
143     printf("El punt inicial 4 és (%lf, %lf) \n", x, y);
144     metode(x,y, fout);
145   }
146
147   fclose(fout);
148
149 return 0;
150 }
```

```
151
152
153 double avaldfx(double x, double y){
154   double dxf;
155
156   dxf = 2*x - (2*(1-mu)*(x-mu))/(sqrt(((x-mu)*(x-mu)+ y*y)*((x-mu)*(x-mu)
      +y*y)*((x-mu)*(x-mu) + y*y)));
157   dxf -= (2*mu*(x-mu+1))/sqrt(((x-mu+1)*(x-mu+1)+ y*y)*((x-mu+1)*(x-mu+1)
      + y*y)*((x-mu+1)*(x-mu+1) + y*y));
158
159   return dxf;
160
161 }
162
163 double avaldfy(double x, double y){
164   double dyf;
165
166     dyf = 2*y -(2*(1-mu)*y)/(sqrt(((x-mu)*(x-mu) + y*y)*((x-mu)*(x-mu) + y
      *y)*((x-mu)*(x-mu) + y*y)));
167     dyf -= (2*mu*y)/(sqrt(((x-mu+1)*(x-mu+1)+ y*y) * ((x-mu+1)*(x-mu+1)+ y
      *y) * ((x-mu+1)*(x-mu+1)+ y*y)));
168   return dyf;
169
170 }
171
172 double avaluaf(double x, double y){
173   double f;
174
175   f = (x*x) + (y*y) - J;
176   f += 2*((1-mu)/(sqrt((x-mu)*(x-mu)+ y*y)) +(mu)/(sqrt((x-mu+1)*(x-mu+1)
      + y*y))) + mu*(1-mu);
177
178   return f;
179 }
180
181 void correccio(double x0, double y0, double *x, double *y){
182   int i = 0;
183   double DF[2][2], F[2], det, dist, aux;
184
185   F[0] = avaluaf(*x, *y);
186   F[1] = ((*x - x0)* (*x -x0)) + ((*y - y0)*(*y-y0)) - (h*h);
187
188   do{
189     DF[0][0] = 2 * (*y - y0);
190     DF[1][0] = (-2) * (*x - x0);
191     DF[0][1] = (-1) * avaldfy(*x, *y);
192     DF[1][1] = avaldfx(*x, *y);
193
194     det = 1./(DF[0][0]*DF[1][1]-DF[1][0]*DF[0][1]);
195
196     aux = *x - (det * DF[0][0]*F[0]) - (det*DF[0][1]*F[1]);
197     *x = aux;
198
199     aux = *y - (det*DF[1][0]*F[0]) - (det*DF[1][1]*F[1]);
200     *y = aux;
201
202     F[0] = avaluaf(*x, *y);
203     F[1] = ((*x - x0)* (*x -x0)) + ((*y - y0)*(*y-y0)) - (h*h);
204
205     dist = sqrt(F[0]*F[0] + F[1]*F[1]);
206     i++;
```

```
207
208
209   }while(i < 100 &&  error < dist);
210
211
212 }
213
214 void metode(double x, double y, FILE* fout){
215   /*Aquí realitzo el mètode predictor corrector*/
216   int i = 0;
217   double x0, y0, dxf, dyf, v[2], norma;
218
219   do{
220     /*Predicció*/
221     x0 = x;
222     y0 = y;
223
224     /*Vectors tangents unitaris*/
225     dxf = avaldfx(x, y);
226     dyf = avaldfy(x, y);
227     v[0] = (-1) * dyf;
228     v[1] = dxf;
229     norma = sqrt(dxf*dxf + dyf*dyf);
230     if(norma < tol){
231       printf("La norma a la predicció és molt petita \n");
232       exit(1);
233     }
234
235     v[0] = v[0]/norma;
236     v[1] = v[1]/norma;
237
238     /*Pas de predicció*/
239
240     x = x + h*v[0];
241     y = y + h*v[1];
242
243     /*Correcció*/
244
245     correccio(x0, y0, &x, &y);
246
247     fprintf(fout, "%11.4le %11.4le\n", x, y);
248     i++;
249
250   }while(i < n);
251 }
```

A continuació adjuntarem diversos codis emprats en la cerca d'òrbites periòdiques. La idea darrera la implementació d'aquests està descrita al llarg de la secció **4**.

Començarem amb l'integrador RK45F:

```
1  /*Autor: Arnau Ruiz Sebastián*/
2  /*Funció rkf45*/
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <math.h>
7
8  #define E1 (1.e0/360.e0)
9  #define E2 (-128.e0/4275.e0)
10 #define E3 (-2197.e0/75240.e0)
```

```c
#define E4 (1.e0/50.e0)
#define E5 (2.e0/55.e0)
#define R1 (16.e0/135.e0)
#define R2 (6656.e0/12825.e0)
#define R3 (28561.e0/56430.e0)
#define R4 (-9.e0/50.e0)
#define R5 (2.e0/55.e0)
#define C1 (12.e0/13.e0)
#define C2 (1932.e0/2197.e0)
#define C3 (-7200.e0/2197.e0)
#define C4 (7296.e0/2197.e0)
#define C5 (439.e0/216.e0)
#define C6 (3680.e0/513.e0)
#define C7 (-845.e0/4104.e0)
#define C8 (-8.e0/27.e0)
#define C9 (-3544.e0/2565.e0)
#define C10 (1859.e0/4104.e0)

#define FS 0.90 //Factor de seguretat pel control de pas


int rk45fC (double *t, double *x, int n, double *h, int sc, double tol,
    double *atf, double *aer, void(*camp)(double, double*, int, double*,
    double e), double e){
  double *k, *k1, *k2, *k3, *k4, *k5, *k6, error, aux;
  int i, rv=0;
  void calcular_ks(double x, double y[], int n, double h, double *k, void
    (*camp)(double, double*, int, double*, double), double e); //Funció que
     calcula les constants k

  k=(double *)malloc(6*n*sizeof(double));
  if (k==NULL) {puts("rk45f. No hi ha memoria per les k's\n");exit(1);}
  k1=k; k2=k1+n; k3=k2+n; k4=k3+n; k5=k4+n; k6=k5+n;

  /*
    Si atf==NULL -> no fem res
    Si atf!=NULL -> Si t+ah>atf llavors reduïm ah per tal que el nostre
    pas sigui de temps atf com volem.
  */
  /*Primer mirem que el temps del nostre pas concorda amb el temps final
    indicat*/

  if(atf!=NULL && *t+*h>*atf){
    *h=*atf-*t; /*reduïm el pas ah tal que el temps final serà atf*/
    rv=1;
  }
  calcular_ks(*t,x,n, *h, k, camp, e);
  error=0.e0;
  for (i=0; i<n; i++){
    aux=E1*k1[i]+E2*k3[i]+E3*k4[i]+E4*k5[i]+E5*k6[i];
    error+=aux*aux;
  }
  error=sqrt(error);

  /*
    Si sc==0 -> no fem control de pas, utilitzem *h
    Si sc!=0 -> control de pas, segons la tolerància
  */
  while(sc!=0 && error>tol){
    /*Control de pas*/
    *h=*h*FS*(pow(tol/error,0.2e0));
```

```
66      calcular_ks(*t,x,n, *h, k, camp, e);
67      error=0.e0;
68      for (i=0; i<n; i++){
69        aux=E1*k1[i]+E2*k3[i]+E3*k4[i]+E4*k5[i]+E5*k6[i];
70        error+=aux*aux;
71      }
72      error=sqrt(error);
73    }
74
75
76    /*
77      Si aer==NULL && error>tol -> exit
78      Si aer!=NULL -> return the estimated absolute error
79
80    */
81    if(aer==NULL && error>tol){
82      printf("Valor aer NULL i error>tol \n");
83      exit(1);
84    }else{
85      /*return the estimated absolute error*/
86      *aer=error;
87    }
88    /*Actualitzem el temps*/
89    *t=*t+*h;
90    for (i=0; i<n; i++){
91      x[i]=x[i]+R1*k1[i]+R2*k3[i]+R3*k4[i]+R4*k5[i]+R5*k6[i]; //Actualitzo
        el pas de Runge-Kutta 5
92    }
93    *h=*h*FS*(pow(tol/error,0.2e0));
94    free(k);
95    return rv;
96  }
97
98  /*El càlcul de les k's*/
99  void calcular_ks (double t, double x[], int n, double h, double *k, void
        (*camp)(double, double*, int, double*, double), double e){
100   double *aux_x, *k1, *k2, *k3, *k4, *k5, *k6;
101   int i;
102   k1=k; k2=k1+n; k3=k2+n; k4=k3+n; k5=k4+n; k6=k5+n;
103   aux_x=(double *)malloc(n*sizeof(double));
104   if (aux_x==NULL) {puts ("calcular_ks. No hi ha memoria \n");exit(1);}
105   (*camp) (t, x, n, k1, e);
106   for (i=0; i<n; i++) k1[i]*=h;
107   for (i=0; i<n; i++) aux_x[i]=x[i]+0.25e0*k1[i];
108   (*camp) (t+0.25e0*h, aux_x, n, k2, e);
109   for (i=0; i<n; i++) k2[i]*=h;
110   for (i=0; i<n; i++) aux_x[i]=x[i]+0.09375e0*k1[i]+0.28125e0*k2[i];
111   (*camp) (t+0.375e0*h, aux_x, n, k3, e);
112   for (i=0; i<n; i++) k3[i]*=h;
113   for (i=0; i<n; i++) aux_x[i]=x[i]+C2*k1[i]+C3*k2[i]+C4*k3[i];
114   (*camp) (t+C1*h, aux_x, n, k4, e);
115   for (i=0; i<n; i++) k4[i]*=h;
116   for (i=0; i<n; i++) aux_x[i]=x[i]+C5*k1[i]-8.e0*k2[i]+C6*k3[i]+C7*k4[i];
117   (*camp) (t+h, aux_x, n, k5, e);
118   for (i=0; i<n; i++) k5[i]*=h;
119   for (i=0; i<n; i++) aux_x[i]=x[i]+C8*k1[i]+2.e0*k2[i]+C9*k3[i]+C10*k4[i
        ]-0.275e0*k5[i];
120   (*camp) (t+0.5e0*h, aux_x, n, k6, e);
121   for (i=0; i<n; i++) k6[i]*=h;
122   free(aux_x);
123 }
```

Ara sí, presentarem el codi que s'utilitza per trobar la primera òrbita periòdica al voltant del punt L_4. Aquest utilitza l'algoritme d'òrbites periòdiques, però no en fa servir encara el mètode de continuació.

```
/* Autor: Arnau Ruiz Sebastián*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>
#include "solis.h"
#include "vaive.h"


#define mu 0.012505819187 //Mu Terra-Lluna
#define as 388.8111430233 //Distància mitja
#define ws 0.925195985518 //Freqüència Sol-Terra-Lluna
#define ns 0.0748040144814 //Moviment mig del Sol respecte el sistema
    Terra-Lluna
#define ms 328900.549999 // Massa del Sol
#define e 1 //Èpsilon
#define maxIt 10 //Iteracions màximes pel mètode de Newton
#define Error 1e-10 //Error del mètode de Newton


int main()
{
  int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
    double *atf, double *aer, void(*camp)(double, double*, int, double*));
  void camp(double t, double x[], int n, double y[]);
  void dist(double x[], double r[], double t);
  void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
     t);
  void newton(double *x, double y[], int n, void(*camp)(double, double*,
    int, double*));

  //Declaració de variables
  char c = 'c';
  int n, sc, j, k, nv;
  double t, x[20], h, atf, aer, tol, **D, mod[4];
  double complex *vap, **vep;
  FILE* fp;

  D = (double **)malloc(4*sizeof(double*));
  if(D == NULL){
    printf("No hi ha memòria suficient \n");
    exit(1);
  }
  for(j=0; j<4; j++){
    D[j] = (double * )malloc(4*sizeof(double));

    if(D[j] == NULL){
      printf("No hi ha memòria suficient \n");
      exit(2);
    }
  }

  vap = (double complex*)malloc(4*sizeof(double complex));
  if(vap == NULL){
    printf("No hi ha memòria suficient \n");
    exit(3);
```

```
54  }
55  vep = (double complex**)malloc(4*sizeof(double complex*));
56  if(vep == NULL){
57    printf("No hi ha memòria suficient \n");
58    exit(4);
59  }
60  for(j=0; j<4; j++){
61    vep[j] = (double complex* )malloc(4*sizeof(double complex));
62
63    if(vep[j] == NULL){
64      printf("No hi ha memòria suficient \n");
65      exit(5);
66    }
67  }
68
69  /*Inicialitzem el temps incial i la dimensió del sistema*/
70  t=0.e0;
71  n=20;
72
73  /*Posem el punt de sortida L4*/
74  x[0]= mu - 0.5;
75  x[1]=sqrt(3)/2;
76  x[2]=0.;
77  x[3]=0.;
78
79  x[4] = 1.;
80  x[5] = 0.;
81  x[6] = 0.;
82  x[7] = 0.;
83
84  x[8] = 0.;
85  x[9] = 1.;
86  x[10] = 0.;
87  x[11] = 0.;
88
89  x[12] = 0.;
90  x[13] = 0.;
91  x[14] = 1.;
92  x[15] = 0.;
93
94  x[16] = 0.;
95  x[17] = 0.;
96  x[18] = 0.;
97  x[19] = 1.;
98
99  /* Newton*/
100 newton(&t, x, n, camp);
101
102 /*Veps i Vaps*/
103 for(k=0; k<4; k++){
104   for(j=0; j<4; j++){
105     D[k][j] = x[4*(k + 1) + j]; //Components de la matriu diferencial
106   }
107 }
108
109 nv = vaive(D, 4, vap, vep, c);//Obtenim els veps i vaps amb un
      programari extern
110
111 printf("La matriu diferencial té %d valors propis reals\n", nv);
112
113 printf("Valors propis complexos:\n");
```

```
114
115   for(j=0; j<4; j++){
116     mod[j] = sqrt( creal(vap[j])*creal(vap[j]) + cimag(vap[j])*cimag(vap[j
      ]) );
117     printf("Vap = %1.10le+%11.4lei  amb mòdul = %1.10le \n", creal(vap[j])
      , cimag(vap[j]), mod[j]);
118   }
119
120   printf("Multiplicació dels mòdul = %e \n", (mod[2]*mod[3] - 1));
121
122   /*Dibuixem l'òrbita*/
123   fp=fopen("bici_op.txt", "w");
124
125   t=0.e0;
126   sc=1; //Volem control de pas
127   atf=(2*M_PI)/ws; //Temps final
128   aer= 1; // Si aer no és NULL retornem el valor absolut de l'error
129   tol=1.e-12;
130   h=1.e-5; //Pas inicial
131
132   fprintf(fp, "%lf %lf %lf %lf %lf \n", t, x[0], x[1], x[2], x[3]);
133   j = 0;
134
135   while (!(rk45f(&t, x, 4, &h, sc, tol, &atf, &aer, camp))){
136     fprintf(fp, "%lf %lf %lf %lf %lf \n", t, x[0], x[1], x[2], x[3]); //
      Dibuixem l'orbita
137   }
138
139   fclose(fp);
140   /*Alliberem memòria*/
141   for(j=0; j<4; j++){
142     free(D[j]);
143   }
144   free(D);
145
146   return 0;
147 }
148
149 void camp(double t, double x[], int n, double f[]){
150   void dist(double x[], double r[], double t);
151   void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
       t);
152   /*Edo per calcular el camp del problema bicircular*/
153   /*x[] = (x, y, x', y')*/
154   double rpt, rpl, rps, r[3], mv[4][4];
155
156   /*Primer calculem les distàncies*/
157   dist(x, r, t);
158
159   rpt = r[0];
160   rpl = r[1];
161   rps = r[2];
162
163
164   /*Camp de l'òrbita*/
165   f[0]= x[2];
166   f[1]= x[3];
167   f[2]= 2* x[3] + x[0] - ((1 - mu)/(rpt*rpt*rpt))*(x[0] - mu) - ((mu)/(rpl
      *rpl*rpl))*(x[0] - mu + 1);
168    f[2] += -e*(ms*(x[0] -as*cos(t-(ns*t)))/(rps*rps*rps)) - e*((ms/(as*as)
      )*cos(t- (ns*t)));
```

```
169   f[3]= -2 * x[2] + x[1] - ((1 - mu)/(rpt*rpt*rpt))*x[1] - ((mu)/(rpl*rpl*
        rpl))*x[1];
170    f[3] += -e *(ms*(x[1] +as*sin(t-(ns*t)))/(rps*rps*rps)) + e*((ms/(as*as
       ))*sin(t- (ns*t)));
171
172    if(n == 4){
173     return;
174    }
175   /*Aportació de les variacionals*/
176   inicialitzarmatriu(mv, r, x, t); /*Ara inicialitzem la matriu
        diferencial del sistema*/
177
178   f[4] = x[12];
179   f[5] = x[13];
180   f[6] = x[14];
181   f[7] = x[15];
182
183   f[8] = x[16];
184   f[9] = x[17];
185   f[10] = x[18];
186   f[11] = x[19];
187
188   f[12] = mv[2][0] * x[4] + mv[2][1] * x[8] + 2* x[16];
189   f[13] = mv[2][0] * x[5] + mv[2][1] * x[9] + 2* x[17];
190   f[14] = mv[2][0] * x[6] + mv[2][1] * x[10]  + 2* x[18];
191   f[15] = mv[2][0] * x[7] + mv[2][1] * x[11] + 2* x[19];
192
193   f[16] = mv[3][0] * x[4] + mv[3][1] * x[8] - 2* x[12];
194   f[17] = mv[3][0] * x[5] + mv[3][1] * x[9] - 2* x[13];
195   f[18] = mv[3][0] * x[6] + mv[3][1] * x[10] - 2* x[14];
196   f[19] = mv[3][0] * x[7] + mv[3][1] * x[11] - 2* x[15];
197
198 }
199
200 void dist(double x[], double r[3], double t){
201   /*Funció per a calcular les distàncies rpt, rpl i rps*/
202
203   r[0] = sqrt((x[0] - mu)*(x[0] - mu) + (x[1]*x[1]));
204   r[1] = sqrt((x[0] - mu + 1)*(x[0] - mu + 1) + (x[1]*x[1]));
205   r[2] = sqrt((x[0] - as*cos(t-(ns*t)))*(x[0] - as*cos(t-(ns*t))) + (x[1]
        + as*sin(t-(ns*t)))*(x[1] + as*sin(t-(ns*t))));
206 }
207
208 void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double t
       ){
209
210   double rpt, rpl, rps, rpt_5, rpl_5, rps_5;
211
212
213   rpt = r[0];
214   rpl = r[1];
215   rps = r[2];
216
217   rpt_5 = rpt*rpt*rpt*rpt*rpt;
218   rpl_5 = rpl*rpl*rpl*rpl*rpl;
219   rps_5 = rps*rps*rps*rps*rps;
220
221
222   /*Omplo la matriu v del sistema variacional*/
223
224   mv[0][0] = 0;
```

```
  mv[0][1] = 0;
  mv[0][2] = 1;
  mv[0][3] = 0;

  mv[1][0] = 0;
  mv[1][1] = 0;
  mv[1][2] = 0;
  mv[1][3] = 1;

  mv[2][0] = 1 - ((1- mu)*(rpt*rpt - 3*(x[0]- mu)*(x[0]-mu))/(rpt_5)) -
    (((mu)*(rpl*rpl -3*(x[0] - mu + 1)*(x[0] - mu + 1)))/(rpl_5));
  mv[2][0] += - ( e*(ms) * (rps*rps - 3 * (x[0] - as*cos(t - (ns*t))) * (x
    [0] - as*cos(t - (ns*t))))  /(rps_5) ); // df1/dx

  mv[2][1] = ((1 - mu)*(x[0] - mu) * 3 * x[1])/(rpt_5) + (3 * mu * x[1] *
    (x[0] - mu + 1) )/(rpl_5);
  mv[2][1] +=  ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(
    t - (ns*t))))/(rps_5)); //df1/dy


  mv[2][2] = 0;
  mv[2][3] = 2;

  mv[3][0] = + ((3 * (1- mu) * x[1] * (x[0] - mu))/(rpt_5)) + (3 * mu * x
    [1] * (x[0] - mu + 1))/(rpl_5);
  mv[3][0] += ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(t
     - (ns*t))))/(rps_5));  //df2/dx

  mv[3][1] = 1 - ((1-mu) * ((rpt*rpt) - (3 * x[1] * x[1]))/(rpt_5)) - (((
    mu) * ((rpl * rpl) - (3 * x[1] * x[1])))/(rpl_5));
  mv[3][1] -=  (((e*ms) * (rps * rps - 3 * (x[1] + as*sin(t - (ns*t))) * (
    x[1] + as*sin(t - (ns*t)))))/ (rps_5));   //df2/dy


  mv[3][2] = -2;
  mv[3][3] = 0;
}


void newton(double *t, double x[], int n, void(*camp)(double, double*, int
    , double*)){

  int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
    double *atf, double *aer, void(*camp)(double, double*, int, double*));


  /*Suposem que tenim la diferencial*/
  double **A, *b, *d, z[4], *aux, error = 1, h, atf, aer, tol;
  int i = 0, j, k, sc;

  /*Variables pel runge-kutta*/

  sc=1;
  atf=(2*M_PI)/ws;
  aer= 1;
  tol=1.e-12;
  h=1.e-5;

  /*Inicialitzem els punters del sistema lineal Ax = b i auxiliars*/

  A = (double **)malloc(4*sizeof(double*));
```

```
276   if(A == NULL){
277     printf("No hi ha memòria suficient \n");
278     exit(1);
279   }
280   for(j=0; j<4; j++){
281     A[j] = (double * )malloc(4*sizeof(double));
282
283     if(A[j] == NULL){
284     printf("No hi ha memòria suficient \n");
285     exit(2);
286     }
287   }
288
289   aux = (double *)malloc(4*sizeof(double));
290   if(x == NULL){
291     printf("No hi ha memòria suficient \n");
292     exit(3);
293   }
294
295   b = (double *)malloc(4*sizeof(double));
296   if(x == NULL){
297     printf("No hi ha memòria suficient \n");
298     exit(4);
299   }
300
301   d = (double *)malloc(sizeof(double));
302   if(d == NULL){
303     printf("No hi ha memòria suficient \n");
304     exit(5);
305   }
306
307   while(fabs(error) > Error && i < maxIt){
308     /*a és un vector de quatre components que és la solució del sistema DF
      (z0)a = -F(z0)*/
309     z[0] = x[0];
310     z[1] = x[1];
311     z[2] = x[2];
312     z[3] = x[3];
313
314     *t=0.e0; //Actualitzem el temps per calcular la imatge de P(x)
315     x[4] = 1.;
316     x[5] = 0.;
317     x[6] = 0.;
318     x[7] = 0.;
319
320     x[8] = 0.;
321     x[9] = 1.;
322     x[10] = 0.;
323     x[11] = 0.;
324
325     x[12] = 0.;
326     x[13] = 0.;
327     x[14] = 1.;
328     x[15] = 0.;
329
330     x[16] = 0.;
331     x[17] = 0.;
332     x[18] = 0.;
333     x[19] = 1.;
334
335     /*Obtenim la imatge de P(x) mitjançant el flux*/
```

```
336     while (!(rk45f(t, x, n, &h, sc, tol, &atf, &aer, camp))){
337     }
338 
339     /*Escrivim les components obtingudes de x al nostre sistema*/
340 
341     for(j=0; j<4; j++){
342       b[j] = - x[j] + z[j]; // G(z) = F(z) - z, volem amb el signe canviat
       pel mètode de Newton
343     }
344     error =  sqrt((b[0] * b[0]) + (b[1]*b[1]) + (b[2]*b[2]) + (b[3]*b[3]))
      ;
345 
346     for(k=0; k<4; k++){
347       for(j=0; j<4; j++){
348         A[k][j] = x[4*(k + 1) + j]; //Components de la matriu diferencial
      DP(x)
349       }
350     }
351 
352     /*Hem de restar la identitat a la nostra matriu diferencial*/
353     A[0][0] = A[0][0] - 1;
354     A[1][1] = A[1][1] - 1;
355     A[2][2] = A[2][2] - 1;
356     A[3][3] = A[3][3] - 1;
357 
358     solis(A, aux, b, 4, d);
359     printf("Estic dins del Newton i = %d determinant %e \n", i, *d);
360     /*Actualitzem la iteració de newton*/
361     x[0] = z[0] + aux[0];
362     x[1] = z[1] + aux[1];
363     x[2] = z[2] + aux[2];
364     x[3] = z[3] + aux[3];
365 
366     i++;
367   }
368 
369   /*Free*/
370   free(aux);
371   free(b);
372   free(d);
373   for(j=0; j<4; j++){
374     free(A[j]);
375   }
376   free(A);
377 
378 }
```

Continuem ara amb el codi emprat per trobar la segona òrbita periòdica en el punt L_4. Cal esmentar que per trobar-ne la tercera només s'ha de realitzar uns petits canvis en el bucle principal corresponent al mètode de continuació.

```
1  /*Autor: Arnau Ruiz Sebastián*/
2 
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <math.h>
6  #include <complex.h>
7  #include "solis.h"
8  #include "vaive.h"
9 
10 
```

```c
#define mu 0.012505819187 //Mu Terra-Lluna
#define as 388.8111430233 //Distància mitja
#define ws 0.925195985518 //Freqüència Sol-Terra-Lluna
#define ns 0.0748040144814 //Moviment mig del Sol respecte el sistema
    Terra-Lluna
#define ms 328900.549999 // Massa del Sol
#define maxIt 10 //Iteracions màximes del mètode de Newton
#define Error 1e-10 //Error del mètode de Newton

double e;

int main()
{
  int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
    double *atf, double *aer, void(*camp)(double, double*, int, double*));
  void camp(double t, double x[], int n, double y[]);
  void dist(double x[], double r[], double t);
  void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
     t);
  int newton(double *x, double y[], int n, void(*camp)(double, double*,
    int, double*), double delta, double y1[5]);
  void omplir(double *x);
  void newtonFirst(double y[], int n, void(*camp)(double, double*, int,
    double*));

  //Inialitzem les variables
  char c = 'c';
  int n, sc, j, k, nv, cont = 0;
  double t, x[24], h, atf, aer, tol, **D, mod[4], y0[5], y1[5], v[5],
    norma, delta = 0.001, aux[5];
  double complex *vap, **vep;
  FILE* fp;

  D = (double **)malloc(4*sizeof(double*));
  if(D == NULL){
    printf("No hi ha memòria suficient \n");
    exit(1);
  }
  for(j=0; j<4; j++){
    D[j] = (double * )malloc(4*sizeof(double));

    if(D[j] == NULL){
      printf("No hi ha memòria suficient \n");
      exit(2);
    }
  }

  vap = (double complex*)malloc(4*sizeof(double complex));
  if(vap == NULL){
    printf("No hi ha memòria suficient \n");
    exit(3);
  }
  vep = (double complex**)malloc(4*sizeof(double complex*));
  if(vep == NULL){
    printf("No hi ha memòria suficient \n");
    exit(4);
  }
  for(j=0; j<4; j++){
    vep[j] = (double complex* )malloc(4*sizeof(double complex));

    if(vep[j] == NULL){
```

```
66        printf("No hi ha memòria suficient \n");
67        exit(5);
68      }
69    }
70
71    /*Primer punt, el propi L4*/
72
73    y0[0] = mu - 0.5;
74    y0[1] = sqrt(3)/2;
75    y0[2] = 0.;
76    y0[3] = 0.;
77
78    y0[4] = 0.; //Epsilon inicial
79
80    /*Segon punt, Newton primer cop*/
81    /*Inicialitzem el temps incial i la dimensió del sistema*/
82    n=24;
83
84    x[0]= mu - 0.5;
85    x[1]=sqrt(3)/2;
86    x[2]=0.;
87    x[3]=0.;
88
89    omplir(x);
90
91    e = delta;
92
93    newtonFirst(x, 20, camp);
94
95    y1[0] = x[0];
96    y1[1] = x[1];
97    y1[2] = x[2];
98    y1[3] = x[3];
99
100   y1[4] = e; //Epsilon segon pas, e = c; amb c molt petita
101
102   /*Fem continuació*/
103   while(e <= 1){
104
105     /*Calculem v*/
106
107     for(k=0; k<5; k++){
108       v[k] = (y1[k] - y0[k]);
109     }
110
111     norma = sqrt( (v[0]*v[0]) + (v[1]*v[1]) + (v[2]*v[2]) + (v[3]*v[3]) +
      (v[4]*v[4]));
112
113     for(k=0; k<5; k++){
114       v[k] = v[k]/norma;
115     }
116
117     //Ens guardem en una variable auxiliar la iteració
118
119     for(k=0; k<4; k++){
120       aux[k] = x[k];
121     }
122     aux[4] = e;
123
124     //Actualitzem la condició incial per Newton
125     for(k=0; k<4; k++){
```

```
126       x[k] += delta*v[k];
127     }
128     e = e + delta*v[4];
129 
130     omplir(x);
131     t=0.e0;
132 
133     printf("Nova iteració de continuació e = %11.4le\n", e);
134 
135     while(newton(&t, x, n, camp, delta, y1)){
136       //Desfem
137       for(k=0; k<4; k++){
138         x[k] = aux[k];
139       }
140       e = aux[4];
141 
142       //Reduim el pas delta i tornem a iterar el newton
143       delta = delta/2;
144       for(k=0; k<4; k++){
145         x[k] += delta*v[k];
146       }
147       e = e + delta*v[4];
148 
149       omplir(x);
150       t=0.e0;
151     }
152 
153     //Actualitzo la iteració
154     for(k=0; k<5; k++){
155       y0[k] = y1[k];
156     }
157     for(k=0; k<4; k++){
158       y1[k] = x[k];
159     }
160     y1[4] = e;
161 
162   }
163 
164   //Hem acabat la continuació
165 
166   /*Condició inicial de l'òrbita periòdica*/
167 
168   printf("Després del Newton x=%11.4le y=%11.4le x'=%11.4le y'=%11.5le \n"
        , x[0], x[1], x[2], x[3]);
169 
170   /*Veps i Vaps*/
171   for(k=0; k<4; k++){
172     for(j=0; j<4; j++){
173       D[k][j] = x[4*(k + 1) + j]; //Components de la matriu diferencial
174     }
175   }
176 
177 
178   nv = vaive(D, 4, vap, vep, c);
179 
180   printf("La matriu diferencial té %d valors propis reals\n", nv);
181 
182   printf("Valors propis complexos:\n");
183 
184   for(j=0; j<4; j++){
185     mod[j] = sqrt( creal(vap[j])*creal(vap[j]) + cimag(vap[j])*cimag(vap[j
```

```
    ]) );
186   printf("Vap = %1.10le+%11.4lei  amb mòdul = %1.10le \n", creal(vap[j])
      , cimag(vap[j]), mod[j]);
187 }
188
189 printf("Multiplicació dels mòdul = %e \n", (mod[2]*mod[3] - 1));
190
191 /*Dibuixem l'òrbita*/
192 fp=fopen("bici_op.txt", "w");
193
194 t=0.e0;
195 sc=1;
196 atf=(2*M_PI)/ws;
197 aer= 1;
198 tol=1.e-12;
199 h=1.e-5;
200
201 fprintf(fp, "%lf %lf %lf %lf %lf \n", t, x[0], x[1], x[2], x[3]);
202
203 while (!(rk45f(&t, x, 4, &h, sc, tol, &atf, &aer, camp))){
204   fprintf(fp, "%lf %lf %lf %lf %lf \n", t, x[0], x[1], x[2], x[3]);
205 }
206
207 fclose(fp);
208 /*Alliberem memòria*/
209 for(j=0; j<4; j++){
210   free(D[j]);
211 }
212 free(D);
213
214 return 0;
215 }
216
217 void camp(double t, double x[], int n, double f[]){
218 void dist(double x[], double r[], double t);
219 void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
      t);
220 /*Edo per calcular el camp del problema bicircular*/
221 /*x[] = (x, y, x', y')*/
222 double rpt, rpl, rps, r[3], mv[4][4];
223
224 /*Primer calculem les distàncies*/
225 dist(x, r, t);
226 rpt = r[0];
227 rpl = r[1];
228 rps = r[2];
229
230 /*Camp de l'òrbita*/
231 f[0]= x[2];
232 f[1]= x[3];
233 f[2]= 2* x[3] + x[0] - ((1 - mu)/(rpt*rpt*rpt))*(x[0] - mu) - ((mu)/(rpl
    *rpl*rpl))*(x[0] - mu + 1);
234   f[2] += -e*(ms*(x[0] -as*cos(t-(ns*t)))/(rps*rps*rps)) - e*((ms/(as*as
    ))*cos(t- (ns*t)));
235 f[3]= -2 * x[2] + x[1] - ((1 - mu)/(rpt*rpt*rpt))*x[1] - ((mu)/(rpl*rpl*
    rpl))*x[1];
236   f[3] += -e *(ms*(x[1] +as*sin(t-(ns*t)))/(rps*rps*rps)) + e*((ms/(as*
    as))*sin(t- (ns*t)));
237
238  if(n == 4){
239   return;
```

```
240     }
241   /*Aportació de les variacionals   V' = mv*V */
242   inicialitzarmatriu(mv, r, x, t); /*Ara inicialitzem la matriu
        diferencial del sistema*/
243
244   f[4] = x[12];
245   f[5] = x[13];
246   f[6] = x[14];
247   f[7] = x[15];
248
249   f[8] = x[16];
250   f[9] = x[17];
251   f[10] = x[18];
252   f[11] = x[19];
253
254   f[12] = mv[2][0] * x[4] + mv[2][1] * x[8] + 2* x[16];
255   f[13] = mv[2][0] * x[5] + mv[2][1] * x[9] + 2* x[17];
256   f[14] = mv[2][0] * x[6] + mv[2][1] * x[10]  + 2* x[18];
257   f[15] = mv[2][0] * x[7] + mv[2][1] * x[11] + 2* x[19];
258
259   f[16] = mv[3][0] * x[4] + mv[3][1] * x[8] - 2* x[12];
260   f[17] = mv[3][0] * x[5] + mv[3][1] * x[9] - 2* x[13];
261   f[18] = mv[3][0] * x[6] + mv[3][1] * x[10] - 2* x[14];
262   f[19] = mv[3][0] * x[7] + mv[3][1] * x[11] - 2* x[15];
263
264   if(n == 20){
265     return;
266   }
267
268   f[20] = x[22];
269   f[21] = x[23];
270   f[22] = mv[2][0] * x[20] + mv[2][1] * x[21] + 2* x[23] -(ms*(x[0] -as*
        cos(t-(ns*t)))/(rps*rps*rps)) - ((ms/(as*as))*cos(t- (ns*t)));
271   f[23] = mv[3][0] * x[20] + mv[3][1] * x[21] - 2* x[22] -(ms*(x[1] +as*
        sin(t-(ns*t)))/(rps*rps*rps)) + ((ms/(as*as))*sin(t- (ns*t)));
272
273 }
274
275 void dist(double x[], double r[3], double t){
276   /*Funció per a calcular les dist?cies rpt, rpl i rps*/
277
278   r[0] = sqrt((x[0] - mu)*(x[0] - mu) + (x[1]*x[1]));
279   r[1] = sqrt((x[0] - mu + 1)*(x[0] - mu + 1) + (x[1]*x[1]));
280   r[2] = sqrt((x[0] - as*cos(t-(ns*t)))*(x[0] - as*cos(t-(ns*t))) + (x[1]
        + as*sin(t-(ns*t)))*(x[1] + as*sin(t-(ns*t))));
281 }
282
283 void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double t
        ){
284
285   double rpt, rpl, rps, rpt_5, rpl_5, rps_5;
286
287   rpt = r[0];
288   rpl = r[1];
289   rps = r[2];
290
291   rpt_5 = rpt*rpt*rpt*rpt*rpt;
292   rpl_5 = rpl*rpl*rpl*rpl*rpl;
293   rps_5 = rps*rps*rps*rps*rps;
294
295   /*Omplo la matriu v del sistema variacional*/
```

```
296
297   mv[0][0] = 0;
298   mv[0][1] = 0;
299   mv[0][2] = 1;
300   mv[0][3] = 0;
301
302   mv[1][0] = 0;
303   mv[1][1] = 0;
304   mv[1][2] = 0;
305   mv[1][3] = 1;
306
307   mv[2][0] = 1 - ((1- mu)*(rpt*rpt - 3*(x[0]- mu)*(x[0]-mu))/(rpt_5)) -
        (((mu)*(rpl*rpl -3*(x[0] - mu + 1)*(x[0] - mu + 1)))/(rpl_5));
308   mv[2][0] += - ( e*(ms) * (rps*rps - 3 * (x[0] - as*cos(t - (ns*t))) * (x
        [0] - as*cos(t - (ns*t))))  /(rps_5) ); // df1/dx
309
310   mv[2][1] = ((1 - mu)*(x[0] - mu) * 3 * x[1])/(rpt_5) + (3 * mu * x[1] *
        (x[0] - mu + 1) )/(rpl_5);
311   mv[2][1] +=  ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(
        t - (ns*t))))/(rps_5)); //df1/dy
312
313
314   mv[2][2] = 0;
315   mv[2][3] = 2;
316
317   mv[3][0] = + ((3 * (1- mu) * x[1] * (x[0] - mu))/(rpt_5)) + (3 * mu * x
        [1] * (x[0] - mu + 1))/(rpl_5);
318   mv[3][0] += ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(t
         - (ns*t))))/(rps_5));  //df2/dx
319
320   mv[3][1] = 1 - ((1-mu) * ((rpt*rpt) - (3 * x[1] * x[1]))/(rpt_5)) - (((
        mu) * ((rpl * rpl) - (3 * x[1] * x[1])))/(rpl_5));
321   mv[3][1] -=  (((e*ms) * (rps * rps - 3 * (x[1] + as*sin(t - (ns*t))) * (
        x[1] + as*sin(t - (ns*t)))))/ (rps_5));   //df2/dy
322
323
324   mv[3][2] = -2;
325   mv[3][3] = 0;
326 }
327
328
329 int newton(double *t, double x[], int n, void(*camp)(double, double*, int,
        double*), double delta, double y1[5]){
330   void omplir(double *x);
331   int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
        double *atf, double *aer, void(*camp)(double, double*, int, double*));
332
333   /*Suposem que tenim la diferencial*/
334   double **A, *b, *d, z[5], *aux, error = 1, h, atf, aer, tol, norma;
335   int i = 0, j, k, sc;
336
337   /*Inicialitzem els punters del sistema lineal Ax = b i auxiliars*/
338
339   A = (double **)malloc(5*sizeof(double*));
340   if(A == NULL){
341     printf("No hi ha memòria suficient \n");
342     exit(1);
343   }
344   for(j=0; j<5; j++){
345     A[j] = (double * )malloc(4*sizeof(double));
346
```

```
347     if(A[j] == NULL){
348     printf("No hi ha memòria suficient \n");
349     exit(2);
350     }
351   }
352
353   aux = (double *)malloc(5*sizeof(double));
354   if(x == NULL){
355     printf("No hi ha memòria suficient \n");
356     exit(3);
357   }
358
359   b = (double *)malloc(5*sizeof(double));
360   if(x == NULL){
361     printf("No hi ha memòria suficient \n");
362     exit(4);
363   }
364
365   d = (double *)malloc(sizeof(double));
366   if(d == NULL){
367     printf("No hi ha memòria suficient \n");
368     exit(5);
369   }
370
371   /*Variables pel runge-kutta*/
372   sc=1;
373   atf=(2*M_PI)/ws;
374   aer= 1;
375     tol=1.e-12;
376   h=1.e-5;
377
378   while(fabs(error) > Error && i < maxIt){
379     /*a és un vector de quatre components que és solució del sistema DF(z0
      )a = -F(z0)*/
380     for(k=0; k<4; k++){
381       z[k] = x[k];
382     }
383     z[4] = e;
384
385     *t=0.e0; //Actualitzem el temps per calcular la imatge de P(x)
386     omplir(x);
387
388     /*Obtenim la imatge de P(x) mitjançant el flux*/
389     while (!(rk45f(t, x, 24, &h, sc, tol, &atf, &aer, camp))){
390     }
391
392     /*Escrivim les components obtingudes de x al nostre sistema*/
393
394     for(j=0; j<4; j++){
395       b[j] = - x[j] + z[j]; // G(z) = F(z) - z, volem amb el signe canviat
       pel mètode de Newton
396     }
397
398     norma = 0.;
399     for(k=0; k<5; k++){
400       norma += (z[k] - y1[k]) * (z[k] - y1[k]);
401     }
402
403     b[4] = - norma + delta*delta; //Corresponent a l'equació de la
      circumferència
404
```

```
405     error =  sqrt((b[0] * b[0]) + (b[1]*b[1]) + (b[2]*b[2]) + (b[3]*b[3]))
        ;
406
407     for(k=0; k<4; k++){
408       for(j=0; j<4; j++){
409         A[k][j] = x[4*(k + 1) + j]; //Components de la matriu diferencial
        DP(x)
410       }
411     }
412     /*Elements de la diferencial degut a l'última equació*/
413     for(j=0; j<4; j++){
414       A[4][j] = 2*(x[j] - y1[j]);
415     }
416     A[4][4] = 2*(e - y1[4]);
417
418     //Resta les derivades de les equacions diferencials respecte el
        paràmetre e
419     A[0][4] = x[20];
420     A[1][4] = x[21];
421     A[2][4] = x[22];
422     A[3][4] = x[23];
423
424
425     /*Hem de restar la identitat a la nostra matriu diferencial*/
426     A[0][0] = A[0][0] - 1;
427     A[1][1] = A[1][1] - 1;
428     A[2][2] = A[2][2] - 1;
429     A[3][3] = A[3][3] - 1;
430
431     solis(A, aux, b, 5, d);
432
433     /*Actualitzem la iteració de Newton*/
434     for(k=0; k<4; k++){
435       x[k] = z[k] + aux[k];
436     }
437     e = e + aux[4];
438
439     i++;
440   }
441
442   /*Free*/
443   free(aux);
444   free(b);
445   free(d);
446   for(j=0; j<4; j++){
447     free(A[j]);
448   }
449   free(A);
450
451   if(i >= maxIt){
452     return 1; //Reduir el pas delta
453   }
454   return 0;
455
456 }
457
458 void omplir(double *x){
459     x[4] = 1.;
460     x[5] = 0.;
461     x[6] = 0.;
462     x[7] = 0.;
```

```
463
464     x[8] = 0.;
465     x[9] = 1.;
466     x[10] = 0.;
467     x[11] = 0.;
468
469     x[12] = 0.;
470     x[13] = 0.;
471     x[14] = 1.;
472     x[15] = 0.;
473
474     x[16] = 0.;
475     x[17] = 0.;
476     x[18] = 0.;
477     x[19] = 1.;
478
479     x[20] = 0.;
480     x[21] = 0.;
481     x[22] = 0.;
482     x[23] = 0.;
483 }
484
485 void newtonFirst(double x[], int n, void(*campFirst)(double, double*, int,
        double*)){
486   void omplir(double *x);
487   int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
        double *atf, double *aer, void(*camp)(double, double*, int, double*));
488
489   /*Suposem que tenim la diferencial*/
490   double **A, *b, *d, z[4], *aux, error = 1, h, atf, aer, tol, distT,
        distL, t;
491   int i = 0, j, k, sc;
492
493   /*Inicialitzem els punters del sistema lineal Ax = b i auxiliars*/
494
495   A = (double **)malloc(4*sizeof(double*));
496   if(A == NULL){
497     printf("No hi ha memòria suficient \n");
498     exit(1);
499   }
500   for(j=0; j<4; j++){
501     A[j] = (double * )malloc(4*sizeof(double));
502
503     if(A[j] == NULL){
504     printf("No hi ha memòria suficient \n");
505     exit(2);
506     }
507   }
508
509   aux = (double *)malloc(4*sizeof(double));
510   if(aux == NULL){
511     printf("No hi ha memòria suficient \n");
512     exit(3);
513   }
514
515   b = (double *)malloc(4*sizeof(double));
516   if(b == NULL){
517     printf("No hi ha memòria suficient \n");
518     exit(4);
519   }
520
```

```
521   d = (double *)malloc(sizeof(double));
522   if(d == NULL){
523     printf("No hi ha memòria suficient \n");
524     exit(5);
525   }
526   /*Variables pel runge-kutta*/
527   sc=1;
528   atf=(2*M_PI)/ws;
529   aer= 1;
530     tol=1.e-12;
531
532   while(fabs(error) > Error && i < maxIt){
533     /*a és un vector de quatre components que és solució del sistema DF(z0
      )a = -F(z0)*/
534     for(k=0; k<4; k++){
535       z[k] = x[k];
536     }
537
538     t=0.e0; //Actualitzem el temps per calcular la imatge de P(x)
539     omplir(x);
540     /*Obtenim la imatge de P(x) mitjançant el flux*/
541     h=1.e-1;
542     while (!(rk45f(&t, x, 20, &h, sc, tol, &atf, &aer, camp))){
543     }
544
545     /*Escrivim les components obtingudes de x al nostre sistema*/
546
547     for(j=0; j<4; j++){
548       b[j] = - x[j] + z[j]; // G(z) = F(z) - z, volem amb el signe canviat
       pel mètode de Newton
549     }
550     error =  sqrt((b[0] * b[0]) + (b[1]*b[1]) + (b[2]*b[2]) + (b[3]*b[3]))
      ;
551          printf("norma funció: %e\n",error);
552
553     for(k=0; k<4; k++){
554       for(j=0; j<4; j++){
555         A[k][j] = x[4*(k + 1) + j]; //Components de la matriu diferencial
      DP(x)
556       }
557     }
558
559     /*Hem de restar la identitat a la nostra matriu diferencial*/
560     A[0][0] = A[0][0] - 1;
561     A[1][1] = A[1][1] - 1;
562     A[2][2] = A[2][2] - 1;
563     A[3][3] = A[3][3] - 1;
564
565     solis(A, aux, b, 4, d);
566
567     /*Actualitzem la iteració de newton*/
568     for(k=0; k<4; k++){
569       x[k] = z[k] + aux[k];
570     }
571
572     i++;
573   }
574
575   /*Free*/
576   free(aux);
577   free(b);
```

```
578   free(d);
579   for(j=0; j<4; j++){
580     free(A[j]);
581   }
582   free(A);
583 
584 }
```

Finalment, presentem l'últim programari emprat al treball. En concret, aquest s'utilitza per a trobar l'òrbita periòdica del punt L_1 presentada a **5.2**.

```
1  /*Autor: Arnau Ruiz Sebastián*/
2 
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <math.h>
6  #include <complex.h>
7  #include "solis.h"
8  #include "vaive.h"
9 
10 
11 #define mu 0.012505819187 //Mu Terra-Lluna
12 #define as 388.8111430233 //Distància mitja
13 #define ws 0.925195985518 //Freqüència Sol-Terra-Lluna
14 #define ns 0.0748040144814 //Moviment mig del Sol respecte el sistema
       Terra-Lluna
15 #define ms 328900.549999 // Massa del Sol
16 #define maxIt 5 //Iteracions maximes del mètode de Newton
17 #define Error 1e-12 //Error del mètode de Newton
18 #define L1 -1.157032814896987 //Punt L1
19 
20 double e;
21 
22 int main()
23 {
24   void omplir(double *x);
25   int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
       double *atf, double *aer, void(*camp)(double, double*, int, double*));
26   void camp(double t, double x[], int n, double y[]);
27   void dist(double x[], double r[], double t);
28   void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
       t);
29   void newtonfirst(double x[], double y[], double t[], double s[], int n,
       void(*camp)(double, double*, int, double*));
30   int newton(double *x, double x0[], double x1[], double x2[], double x3
       [], int n, void(*camp)(double, double*, int, double*), double delta,
       double y1[17]);
31 
32   //Inicitalitzem les variables
33   char c = 'c';
34   int n, sc, j, k, nv, i=0;
35   double tol, aer, h, atf, t0, t1, t2, t3, t4, x[24], y[24], t[24], s[24],
       **D, delta = 0.001, aux[17],  y0[17], y1[17], v[17], norma;
36   double complex *vap, **vep;
37   FILE* fp;
38   FILE* fp2;
39 
40 
41   D = (double **)malloc(4*sizeof(double*));
42   if(D == NULL){
43     printf("No hi ha memòria suficient \n");
```

```
    exit(1);
  }
  for(j=0; j<4; j++){
    D[j] = (double * )malloc(4*sizeof(double));

    if(D[j] == NULL){
      printf("No hi ha memòria suficient \n");
      exit(2);
    }
  }

  vap = (double complex*)malloc(4*sizeof(double complex));
  if(vap == NULL){
    printf("No hi ha memòria suficient \n");
    exit(3);
  }
  vep = (double complex**)malloc(4*sizeof(double complex*));
  if(vep == NULL){
    printf("No hi ha memòria suficient \n");
    exit(4);
  }
  for(j=0; j<4; j++){
    vep[j] = (double complex* )malloc(4*sizeof(double complex));

    if(vep[j] == NULL){
      printf("No hi ha memòria suficient \n");
      exit(5);
    }
  }

  n=24;

  /*Arranquem el mètode de continuació*/

  /*Primer punt*/
  e = 0.0001;
  fp2=fopen("e.txt", "w");
  fprintf(fp2, "%lf %lf \n", x[0], e);
  i++;

  /*Sistema 1*/
  x[0]= L1;
  x[1]=0.;
  x[2]=0.;
  x[3]=0.;
  omplir(x); //Omplim la resta del sistema

  /*Sistema 2 */

  y[0]= L1;
  y[1]= 0.;
  y[2]= 0.;
  y[3]= 0.;
  omplir(y); //Omplim la resta del sistema

  /*Sistema 3 */
  t[0]= L1;
  t[1]= 0.;
  t[2]= 0.;
  t[3]= 0.;
  omplir(t); //Omplim la resta del sistema
```

```
105
106 /*Sistema 4*/
107 s[0]= L1;
108 s[1]=0.;
109 s[2]=0.;
110 s[3]=0.;
111 omplir(s); //Omplim la resta del sistema
112
113 /* Newton*/
114 newtonfirst(x, y, t, s, 20, camp);
115
116 for(k=0; k<4; k++){
117   y0[k] = x[k];
118 }
119 for(k=0; k<4; k++){
120   y0[4+k] = y[k];
121 }
122 for(k=0; k<4; k++){
123   y0[8+k] = t[k];
124 }
125 for(k=0; k<4; k++){
126   y0[12+k] = s[k];
127 }
128 y0[16] = e;
129
130 /*Segon punt */
131 e = e + delta;
132
133 /*Sistema 1*/
134
135 omplir(x); //Omplim la resta del sistema
136
137 /*Sistema 2 */
138
139 omplir(y); //Omplim la resta del sistema
140
141 /*Sistema 3 */
142
143 omplir(t); //Omplim la resta del sistema
144
145 /*Sistema 4*/
146
147 omplir(s); //Omplim la resta del sistema
148
149 /* Newton*/
150 newtonfirst(x, y, t, s, 20, camp);
151
152 for(k=0; k<4; k++){
153   y1[k] = x[k];
154 }
155 for(k=0; k<4; k++){
156   y1[4+k] = y[k];
157 }
158 for(k=0; k<4; k++){
159   y1[8+k] = t[k];
160 }
161 for(k=0; k<4; k++){
162   y1[12+k] = s[k];
163 }
164 y1[16] = e;
165
```

```
166   fprintf(fp2, "%lf %lf \n", x[0], e);
167   i++;
168
169   /*Fem continuació*/
170   while(e <= 1 && e > 0){
171
172     /*Calculem v*/
173     for(j=0; j<17; j++){
174       v[j] = (y1[j] - y0[j]);
175     }
176     norma = 0.0;
177     for(j=0; j<17; j++){
178       norma += (v[j]*v[j]);
179     }
180     norma = sqrt(norma);
181
182     for(j=0; j<17; j++){
183       v[j] = v[j]/norma;
184     }
185
186     //Ens guardem en una variable auxiliar la iteració
187     for(k=0; k<4; k++){
188       aux[k] = x[k];
189     }
190     for(k=0; k<4; k++){
191       aux[4+k] = y[k];
192     }
193     for(k=0; k<4; k++){
194       aux[8+k] = t[k];
195     }
196     for(k=0; k<4; k++){
197       aux[12+k] = s[k];
198     }
199     aux[16] = e;
200
201     //Actualitzem la condició incial per Newton
202     if(e + delta*v[16] <= 1){
203       for(k=0; k<4; k++){
204         x[k] += delta*v[k];
205       }
206       for(k=0; k<4; k++){
207         y[k] += delta*v[4+k];
208       }
209       for(k=0; k<4; k++){
210         t[k] += delta*v[8+k];
211       }
212       for(k=0; k<4; k++){
213         s[k] += delta*v[12+k];
214       }
215       e = e + delta*v[16];
216
217     }else if(v[16] != 0){
218       printf("Pas especial de continuació \n");
219       delta = (1-e)/v[16];
220       for(k=0; k<4; k++){
221         x[k] += delta*v[k];
222       }
223       for(k=0; k<4; k++){
224         y[k] += delta*v[4+k];
225       }
226       for(k=0; k<4; k++){
```

```
227         t[k] += delta*v[8+k];
228       }
229       for(k=0; k<4; k++){
230         s[k] += delta*v[12+k];
231       }
232       e = e + delta*v[16];
233     }
234
235     omplir(x);
236     omplir(y);
237     omplir(t);
238     omplir(s);
239
240     t0=0.e0;
241
242     fprintf(fp2, "%lf %lf \n", x[0], e);
243     i++;
244
245     while(newton(&t0, x, y, t, s, 24, camp, delta, y1)){
246       //Desfem
247       for(k=0; k<4; k++){
248         x[k] = aux[k];
249       }
250       for(k=0; k<4; k++){
251         y[k] = aux[4+k];
252       }
253       for(k=0; k<4; k++){
254         t[k] = aux[8+k];
255       }
256       for(k=0; k<4; k++){
257         s[k] = aux[12+k];
258       }
259       e = aux[16];
260
261       //Reduïm el pas delta i tornem a iterar
262       delta = delta/2;
263       for(k=0; k<4; k++){
264         x[k] += delta*v[k];
265       }
266       for(k=0; k<4; k++){
267         y[k] += delta*v[4+k];
268       }
269       for(k=0; k<4; k++){
270         t[k] += delta*v[8+k];
271       }
272       for(k=0; k<4; k++){
273         s[k] += delta*v[12+k];
274       }
275       e = e + delta*v[16];
276
277       omplir(x);
278       omplir(y);
279       omplir(t);
280       omplir(s);
281       t0=0.e0;
282     }
283     //Actualitzo la iteració
284     for(j=0; j<17; j++){
285       y0[j] = y1[j];
286     }
287
```

```
288     for(j=0; j<4; j++){
289       y1[j] = x[j];
290     }
291     for(j=0; j<4; j++){
292       y1[4+j] = y[j];
293     }
294     for(j=0; j<4; j++){
295       y1[8+j] = t[j];
296     }
297     for(j=0; j<4; j++){
298       y1[12+j] = s[j];
299     }
300     y1[16] = e;
301 
302   }
303   /*Dibuixem l'òrbita*/
304   fp=fopen("tir_op.txt", "w");
305 
306   t0=0.e0;
307   sc=1;
308   atf=(2*M_PI)/ws;
309   tol=1.e-12;
310   h=1.e-5;
311 
312   fprintf(fp, "%lf %lf %lf %lf %lf \n", t0, x[0], x[1], x[2], x[3]);
313   j = 0;
314 
315   while (!(rk45f(&t0, x, 4, &h, sc, tol, &atf, &aer, camp))){
316     fprintf(fp, "%lf %lf %lf %lf %lf \n", t0, x[0], x[1], x[2], x[3]); //
      Dibuixem l'òrbita
317   }
318 
319   /*Alliberem memòria*/
320   fclose(fp);
321   fclose(fp2);
322 
323   return 0;
324 }
325 
326 void camp(double t, double x[], int n, double f[]){
327   void dist(double x[], double r[], double t);
328   void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double
       t);
329   /*Edo per calcular el camp*/
330   /*x[] = (x, y, x', y')*/
331   double rpt, rpl, rps, r[3], mv[4][4];
332 
333   /*Primer calculem les distàncies*/
334   dist(x, r, t);
335 
336   rpt = r[0];
337   rpl = r[1];
338   rps = r[2];
339 
340   /*Camp de l'òrbita*/
341   f[0]= x[2];
342   f[1]= x[3];
343   f[2]= 2* x[3] + x[0] - ((1 - mu)/(rpt*rpt*rpt))*(x[0] - mu) - ((mu)/(rpl
      *rpl*rpl))*(x[0] - mu + 1);
344     f[2] += -e*(ms*(x[0] -as*cos(t-(ns*t)))/(rps*rps*rps)) - e*((ms/(as*as
    ))*cos(t- (ns*t)));
```

```
345   f[3]= -2 * x[2] + x[1] - ((1 - mu)/(rpt*rpt*rpt))*x[1] - ((mu)/(rpl*rpl*
        rpl))*x[1];
346    f[3] += -e *(ms*(x[1] +as*sin(t-(ns*t)))/(rps*rps*rps)) + e*((ms/(as*
        as))*sin(t- (ns*t)));
347
348    if(n == 4){
349     return;
350    }
351   /*Aportació de les variacionals*/
352   inicialitzarmatriu(mv, r, x, t); /*Ara inicialitzem la matriu
        diferencial del sistema*/
353
354   f[4] = x[12];
355   f[5] = x[13];
356   f[6] = x[14];
357   f[7] = x[15];
358
359   f[8] = x[16];
360   f[9] = x[17];
361   f[10] = x[18];
362   f[11] = x[19];
363
364   f[12] = mv[2][0] * x[4] + mv[2][1] * x[8] + 2* x[16];
365   f[13] = mv[2][0] * x[5] + mv[2][1] * x[9] + 2* x[17];
366   f[14] = mv[2][0] * x[6] + mv[2][1] * x[10]  + 2* x[18];
367   f[15] = mv[2][0] * x[7] + mv[2][1] * x[11] + 2* x[19];
368
369   f[16] = mv[3][0] * x[4] + mv[3][1] * x[8] - 2* x[12];
370   f[17] = mv[3][0] * x[5] + mv[3][1] * x[9] - 2* x[13];
371   f[18] = mv[3][0] * x[6] + mv[3][1] * x[10] - 2* x[14];
372   f[19] = mv[3][0] * x[7] + mv[3][1] * x[11] - 2* x[15];
373
374   if(n == 20){return;}
375
376   f[20] = x[22];
377   f[21] = x[23];
378   f[22] = mv[2][0] * x[20] + mv[2][1] * x[21] + 2* x[23] -(ms*(x[0] -as*
        cos(t-(ns*t)))/(rps*rps*rps)) - ((ms/(as*as))*cos(t- (ns*t)));
379   f[23] = mv[3][0] * x[20] + mv[3][1] * x[21] - 2* x[22] -(ms*(x[1] +as*
        sin(t-(ns*t)))/(rps*rps*rps)) + ((ms/(as*as))*sin(t- (ns*t)));
380
381 }
382
383 void dist(double x[], double r[3], double t){
384   /*Funció per a calcular les distàncies rpt, rpl i rps*/
385
386   r[0] = sqrt((x[0] - mu)*(x[0] - mu) + (x[1]*x[1]));
387   r[1] = sqrt((x[0] - mu + 1)*(x[0] - mu + 1) + (x[1]*x[1]));
388   r[2] = sqrt((x[0] - as*cos(t-(ns*t)))*(x[0] - as*cos(t-(ns*t))) + (x[1]
        + as*sin(t-(ns*t)))*(x[1] + as*sin(t-(ns*t))));
389 }
390
391 void inicialitzarmatriu(double mv[4][4], double r[3], double x[], double t
        ){
392
393   double rpt, rpl, rps, rpt_5, rpl_5, rps_5;
394
395   rpt = r[0];
396   rpl = r[1];
397   rps = r[2];
398
```

```
399   rpt_5 = rpt*rpt*rpt*rpt*rpt;
400   rpl_5 = rpl*rpl*rpl*rpl*rpl;
401   rps_5 = rps*rps*rps*rps*rps;
402
403   /*Omplo la matriu v del sistema variacional*/
404
405   mv[0][0] = 0;
406   mv[0][1] = 0;
407   mv[0][2] = 1;
408   mv[0][3] = 0;
409
410   mv[1][0] = 0;
411   mv[1][1] = 0;
412   mv[1][2] = 0;
413   mv[1][3] = 1;
414
415   mv[2][0] = 1 - ((1- mu)*(rpt*rpt - 3*(x[0]- mu)*(x[0]-mu))/(rpt_5)) -
        (((mu)*(rpl*rpl -3*(x[0] - mu + 1)*(x[0] - mu + 1)))/(rpl_5));
416   mv[2][0] += - ( e*(ms) * (rps*rps - 3 * (x[0] - as*cos(t - (ns*t))) * (x
        [0] - as*cos(t - (ns*t))))  /(rps_5) ); // df1/dx
417
418   mv[2][1] = ((1 - mu)*(x[0] - mu) * 3 * x[1])/(rpt_5) + (3 * mu * x[1] *
        (x[0] - mu + 1) )/(rpl_5);
419   mv[2][1] +=  ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(
        t - (ns*t))))/(rps_5)); //df1/dy
420
421
422   mv[2][2] = 0;
423   mv[2][3] = 2;
424
425   mv[3][0] = + ((3 * (1- mu) * x[1] * (x[0] - mu))/(rpt_5)) + (3 * mu * x
        [1] * (x[0] - mu + 1))/(rpl_5);
426   mv[3][0] += ((3 * e* ms * (x[0] - as*cos(t - (ns*t))) * (x[1] + as*sin(t
         - (ns*t))))/(rps_5));  //df2/dx
427
428   mv[3][1] = 1 - ((1-mu) * ((rpt*rpt) - (3 * x[1] * x[1]))/(rpt_5)) - (((
        mu) * ((rpl * rpl) - (3 * x[1] * x[1])))/(rpl_5));
429   mv[3][1] -=  (((e*ms) * (rps * rps - 3 * (x[1] + as*sin(t - (ns*t))) * (
        x[1] + as*sin(t - (ns*t)))))/ (rps_5));   //df2/dy
430
431
432   mv[3][2] = -2;
433   mv[3][3] = 0;
434 }
435
436
437 void newtonfirst(double x[], double y[], double t[], double s[], int n,
        void(*camp)(double, double*, int, double*)){
438   void omplir(double *x);
439   int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
        double *atf, double *aer, void(*camp)(double, double*, int, double*));
440
441
442   /*Suposem que tenim la diferencial*/
443   double **A, *b, *d, z[16], *aux, error = 1, h, atf1, atf2, atf3, atf4,
        aer, tol, t1, t2, t3, t4;
444   int i = 0, j, k, sc;
445   /*Variables pel runge-kutta*/
446   sc=1;
447   atf1=(2*M_PI)/(4*ws);
448   atf2=(2*2*M_PI)/(4*ws);
```

```
449   atf3=(3*2*M_PI)/(4*ws);
450   atf4=(2*M_PI)/(ws);
451   aer= 1;
452   tol=1.e-12;
453   h=1.e-5;
454
455   /*Inicialitzem els punters del sistema lineal Ax = b i auxiliars*/
456
457   A = (double **)malloc(16*sizeof(double*));
458   if(A == NULL){
459     printf("No hi ha memòria suficient \n");
460     exit(1);
461   }
462   for(j=0; j<16; j++){
463     A[j] = (double * )malloc(16*sizeof(double));
464
465     if(A[j] == NULL){
466     printf("No hi ha memòria suficient \n");
467     exit(2);
468     }
469   }
470
471   aux = (double *)malloc(16*sizeof(double));
472   if(x == NULL){
473     printf("No hi ha memòria suficient \n");
474     exit(3);
475   }
476
477   b = (double *)malloc(16*sizeof(double));
478   if(x == NULL){
479     printf("No hi ha memòria suficient \n");
480     exit(4);
481   }
482
483   d = (double *)malloc(sizeof(double));
484   if(d == NULL){
485     printf("No hi ha memòria suficient \n");
486     exit(5);
487   }
488
489   while(fabs(error) > Error && i < maxIt){
490     /*Guardem la iteració x_k i calculem x_(k+1)*/
491     for(j=0; j<4; j++){
492       z[j] = x[j];
493     }
494     for(j=0; j<4; j++){
495       z[4+j] = y[j];
496     }
497     for(j=0; j<4; j++){
498       z[8+j] = t[j];
499     }
500     for(j=0; j<4; j++){
501       z[12+j] = s[j];
502     }
503     //Actualitzem el temps per calcular la imatge de P(x)
504     t1=0.e0;
505     t2=atf1;
506     t3=atf2;
507     t4=atf3;
508     omplir(x);
509     omplir(y);
```

```
510     omplir(t);
511     omplir(s);
512
513     /*Obtenim la imatge de P(x) mitjançant el flux pels 4 sistemes*/
514     while (!(rk45f(&t1, x, 20, &h, sc, tol, &atf1, &aer, camp))){}
515     while (!(rk45f(&t2, y, 20, &h, sc, tol, &atf2, &aer, camp))){}
516     while (!(rk45f(&t3, t, 20, &h, sc, tol, &atf3, &aer, camp))){}
517     while (!(rk45f(&t4, s, 20, &h, sc, tol, &atf4, &aer, camp))){}
518
519     /*Escrivim les components obtingudes de x al nostre sistema*/
520
521     for(j=0; j<4; j++){
522       b[j] = - x[j] + z[4 + j];
523     }
524     for(j=4; j<8; j++){
525       b[j] = - y[(j%4)] + z[8+(j%4)];
526     }
527     for(j=8; j<12; j++){
528       b[j] = - t[(j%4)] + z[12 + (j%4)];
529     }
530     for(j=12; j<16; j++){
531       b[j] = - s[(j%4)] + z[(j%4)];
532     }
533
534     error = 0.;
535     for(j=0; j<16; j++){
536       error +=  (b[j] * b[j]);
537     }
538     error = sqrt(error);
539
540         //Matriu diferencial del multi-shotting
541         for(k=0; k<16; k++){
542       for(j=0; j<16; j++){
543         A[k][j] = 0.; //La omplim de zeros
544       }
545     }
546
547     //Components de la matriu diferencial del primer sistema
548     for(k=0; k<4; k++){
549       for(j=0; j<4; j++){
550         A[k][j] = x[4*(k + 1) + j];
551       }
552     }
553     /* - ID sistema 1*/
554     A[0][4] = - 1;
555     A[1][5] = - 1;
556     A[2][6] = - 1;
557     A[3][7] = - 1;
558
559     //Components de la matriu diferencial del segon sistema
560     for(k=4; k<8; k++){
561       for(j=4; j<8; j++){
562         A[k][j] = y[4*((k%4) + 1) + (j%4)];
563       }
564     }
565     /* - ID sistema 2*/
566     A[4][8] = - 1;
567     A[5][9] =  - 1;
568     A[6][10] =  - 1;
569     A[7][11] =  - 1;
570
```

```
571     //Components de la matriu diferencial del tercer sistema
572     for(k=8; k<12; k++){
573       for(j=8; j<12; j++){
574         A[k][j] = t[4*((k%4) + 1) + (j%4)];
575       }
576     }
577     /* - ID sistema 3*/
578     A[8][12] =  - 1;
579     A[9][13] =  - 1;
580     A[10][14] =  - 1;
581     A[11][15] =  - 1;
582 
583     //Components de la matriu diferencial del quart sistema
584     for(k=12; k<16; k++){
585       for(j=12; j<16; j++){
586         A[k][j] = s[4*((k%4) + 1) + (j%4)];
587       }
588     }
589     /* - ID sistema 4*/
590     A[12][0] =  - 1;
591     A[13][1] =  - 1;
592     A[14][2] =  - 1;
593     A[15][3] =  - 1;
594 
595     solis(A, aux, b, 16, d);
596     printf("Estic dins del newtonfirst i = %d \n", i);
597     /*Actualitzem la iteració de Newton*/
598 
599     for(j=0; j<4; j++){
600       x[j] = z[j] + aux[j];
601     }
602     for(j=0; j<4; j++){
603       y[j] = z[4+j] + aux[4+j];
604     }
605     for(j=0; j<4; j++){
606       t[j] = z[8+j] + aux[8+j];
607     }
608     for(j=0; j<4; j++){
609       s[j] = z[12+j] + aux[12+j];
610     }
611 
612     i++;
613   }
614 
615   /*Free*/
616   free(aux);
617   free(b);
618   free(d);
619   for(j=0; j<16; j++){
620     free(A[j]);
621   }
622   free(A);
623 
624 }
625 
626 void omplir(double *x){
627     x[4] = 1.;
628     x[5] = 0.;
629     x[6] = 0.;
630     x[7] = 0.;
631 
```

```
632     x[8] = 0.;
633     x[9] = 1.;
634     x[10] = 0.;
635     x[11] = 0.;
636 
637     x[12] = 0.;
638     x[13] = 0.;
639     x[14] = 1.;
640     x[15] = 0.;
641 
642     x[16] = 0.;
643     x[17] = 0.;
644     x[18] = 0.;
645     x[19] = 1.;
646 
647     x[20] = 0.;
648     x[21] = 0.;
649     x[22] = 0.;
650     x[23] = 0.;
651 }
652 
653 int newton(double *t0, double x[], double y[], double t[], double s[], int
        n, void(*camp)(double, double*, int, double*), double delta, double y1
    []){
654   void omplir(double *x);
655   int rk45f(double *x, double y[], int n, double *h, int sc, double tol,
      double *atf, double *aer, void(*camp)(double, double*, int, double*));
656 
657   /*Suposem que tenim la diferencial*/
658   double **A, *b, *d, z[17], *aux, error = 1, h, aer, tol, norma, t1, t2,
      t3, t4, atf1, atf2, atf3, atf4;
659   int i = 0, j, k, sc;
660 
661   /*Inicialitzem els punters del sistema lineal Ax = b i auxiliars*/
662 
663   A = (double **)malloc(17*sizeof(double*));
664   if(A == NULL){
665     printf("No hi ha memòria suficient \n");
666     exit(1);
667   }
668   for(j=0; j<17; j++){
669     A[j] = (double * )malloc(17*sizeof(double));
670 
671     if(A[j] == NULL){
672     printf("No hi ha memòria suficient \n");
673     exit(2);
674     }
675   }
676 
677   aux = (double *)malloc(17*sizeof(double));
678   if(x == NULL){
679     printf("No hi ha memòria suficient \n");
680     exit(3);
681   }
682 
683   b = (double *)malloc(17*sizeof(double));
684   if(x == NULL){
685     printf("No hi ha memòria suficient \n");
686     exit(4);
687   }
688 
```

```
689   /*Variables pel runge-kutta*/
690   sc=1;
691   atf1=(2*M_PI)/(4*ws);
692   atf2=(2*2*M_PI)/(4*ws);
693   atf3=(3*2*M_PI)/(4*ws);
694   atf4=(2*M_PI)/(ws);
695   aer= 1;
696     tol=1.e-12;
697   h=1.e-5;
698
699   while(fabs(error) > Error && i < maxIt){
700     /*a és un vector de quatre components que és solució del sistema DF(z0
       )a = -F(z0)*/
701     printf("Iteració i= %d de newton\n", i );
702     for(j=0; j<4; j++){
703       z[j] = x[j];
704     }
705     for(j=0; j<4; j++){
706       z[4+j] = y[j];
707     }
708     for(j=0; j<4; j++){
709       z[8+j] = t[j];
710     }
711     for(j=0; j<4; j++){
712       z[12+j] = s[j];
713     }
714     z[16] = e;
715
716     *t0=0.e0;
717     t2=atf1;
718     t3=atf2;
719     t4=atf3;
720     omplir(x);
721     omplir(y);
722     omplir(t);
723     omplir(s);
724
725     /*Obtenim la imatge de P(x) mitjançant el flux*/
726     while (!(rk45f(t0, x, 24, &h, sc, tol, &atf1, &aer, camp))){}
727     while (!(rk45f(&t2, y, 24, &h, sc, tol, &atf2, &aer, camp))){}
728     while (!(rk45f(&t3, t, 24, &h, sc, tol, &atf3, &aer, camp))){}
729     while (!(rk45f(&t4, s, 24, &h, sc, tol, &atf4, &aer, camp))){}
730
731     for(j=0; j<4; j++){
732       b[j] = - x[j] + z[4 + j];
733     }
734     for(j=4; j<8; j++){
735       b[j] = - y[(j%4)] + z[8+(j%4)];
736     }
737     for(j=8; j<12; j++){
738       b[j] = - t[(j%4)] + z[12 + (j%4)];
739     }
740     for(j=12; j<16; j++){
741       b[j] = - s[(j%4)] + z[(j%4)];
742     }
743
744     //Calculem la norma
745     norma = 0.0;
746     for(k=0; k<17; k++){
747       norma += (z[k]-y1[k])*(z[k]-y1[k]);
748     }
```

```
749     b[16] = - norma + delta*delta; //Corresponent a l'equació de la
        circumferència
750
751     error = 0.0;
752     for(k=0; k<16; k++){
753       error += (b[k] * b[k]);
754     }
755
756     error = sqrt(error);
757
758         //Matriu diferencial
759         for(k=0; k<17; k++){
760       for(j=0; j<17; j++){
761         A[k][j] = 0.;
762       }
763     }
764
765     //Components de la matriu diferencial del primer sistema
766     for(k=0; k<4; k++){
767       for(j=0; j<4; j++){
768         A[k][j] = x[4*(k + 1) + j];
769       }
770     }
771     /* - ID sistema 1*/
772     A[0][4] = - 1;
773     A[1][5] = - 1;
774     A[2][6] = - 1;
775     A[3][7] = - 1;
776
777     //Components de la matriu diferencial del segon sistema
778     for(k=4; k<8; k++){
779       for(j=4; j<8; j++){
780         A[k][j] = y[4*((k%4) + 1) + (j%4)];
781       }
782     }
783     /* - ID sistema 2*/
784     A[4][8] = - 1;
785     A[5][9] =  - 1;
786     A[6][10] =  - 1;
787     A[7][11] =  - 1;
788
789     //Components de la matriu diferencial del tercer sistema
790     for(k=8; k<12; k++){
791       for(j=8; j<12; j++){
792         A[k][j] = t[4*((k%4) + 1) + (j%4)];
793       }
794     }
795     /* - ID sistema 3*/
796     A[8][12] =  - 1;
797     A[9][13] =  - 1;
798     A[10][14] =  - 1;
799     A[11][15] =  - 1;
800
801     //Components de la matriu diferencial del quart sistema
802     for(k=12; k<16; k++){
803       for(j=12; j<16; j++){
804         A[k][j] = s[4*((k%4) + 1) + (j%4)];
805       }
806     }
807     /* - ID sistema 4*/
808     A[12][0] =  - 1;
```

```
809     A[13][1] =  - 1;
810     A[14][2] =  - 1;
811     A[15][3] =  - 1;
812 
813     /*Elements de la diferencial degut a l'última equació*/
814     for(j=0; j<4; j++){
815       A[16][j] = 2*(x[j] - y1[j]);
816     }
817     for(j=4; j<8; j++){
818       A[16][j] = 2*(y[j%4] - y1[j]);
819     }
820     for(j=8; j<12; j++){
821       A[16][j] = 2*(t[j%4] - y1[j]);
822     }
823     for(j=12; j<16; j++){
824       A[16][j] = 2*(s[j%4] - y1[j]);
825     }
826     A[16][16] = 2*(e - y1[16]);
827 
828     //Resta les derivades de les equacions diferencials respecte el
      paràmetre e
829 
830     for(j=0; j<4; j++){
831       A[j][16] = x[20 + j];
832     }
833     for(j=4; j<8; j++){
834       A[j][16] = y[20 + (j%4)];
835     }
836     for(j=8; j<12; j++){
837       A[j][16] = t[20 + (j%4)];
838     }
839     for(j=12; j<16; j++){
840       A[j][16] = s[20 + (j%4)];
841     }
842 
843     solis(A, aux, b, 17, d);
844 
845     /*Actualitzem la iteracio de newton*/
846     for(j=0; j<4; j++){
847       x[j] = z[j] + aux[j];
848     }
849     for(j=4; j<8; j++){
850       y[j%4] = z[j] + aux[j];
851     }
852     for(j=8; j<12; j++){
853       t[j%4] = z[j] + aux[j];
854     }
855     for(j=12; j<16; j++){
856       s[j%4] = z[j] + aux[j];
857     }
858     e = e + aux[16];
859 
860     i++;
861   }
862 
863   /*Free*/
864   free(aux);
865   free(b);
866   for(j=0; j<17; j++){
867     free(A[j]);
868   }
```

```
869   free(A);
870
871   if(i >= maxIt){
872     return 1; //Reduïm el pas delta
873   }
874   return 0;
875 }
```