

## Treball final de grau

**Estudi:** Grau en Tecnologies Industrials

**Títol:** Transport natural interplanetari a partir d'un model restringit de tres cossos

**Document:** Memòria

**Alumne:** Joan Esteba Masjuan

**Tutor:** Esther Barrabés Vera

**Departament:** Informàtica, Matemàtica Aplicada i Estadística

**Àrea:** Matemàtica aplicada

**Convocatòria (mes/any):** 06/2016

## Índex

1 Introducció .....	3
1.1 Antecedents .....	3
1.2 Objecte.....	4
1.3 Especificacions i abast .....	4
2 Model matemàtic .....	5
2.1 Problema restringit a tres cossos .....	5
2.2 Punts d'equilibri .....	8
2.2.1 Punts d'equilibri triangulars .....	10
2.2.2 Punts d'equilibri col·lineals .....	11
2.3 Energia associada.....	12
3 Metodologia tècnica .....	16
3.1 Mètode d'integració numèrica Runge-Kutta-Fehlberg.....	16
3.2 Mètode de Newton-Raphson .....	17
3.3 Càlcul dels punts inicials .....	17
3.4 Conversió del sistema de referència.....	18
4 Plantejament i desenvolupament .....	20
4.1 Plantejament .....	20
4.2 Càlcul de trajectòries d'òrbites.....	21
4.2 Integració a secció .....	22
4.3 Càlcul dels punts inicials .....	22
4.3.1 Càlcul de la posició dels punts.....	23
4.3.2 Càlcul de la velocitat dels punts .....	24
4.4 Càlcul de trajectòries per a varis punts.....	24
4.5 Contrast de dades .....	25
4.5.1 Conversió d'unitats .....	25
4.5.2 Cerca de coincidències .....	25
5 Resultats.....	26
5.1 Anàlisis visual de les òrbites.....	26

5.1.1 Òrbites amb origen $L_3$ de la Terra .....	26
5.1.2 Òrbites amb origen $L_1$ de Mart .....	29
5.2 Exploració amb origen a $L_1$ de Mart.....	31
5.3 Exploració amb origen a $L_3$ de la Terra.....	33
5.4 Conversió d'unitats del sistema Sol-Mart al sistema Sol-Terra .....	34
5.5 Comparació de les dues exploracions .....	35
6 Resum del pressupost .....	37
7 Conclusions .....	38
8 Relació de documents .....	40
9 Bibliografia .....	41
ANNEXOS .....	42
A- Codi informàtic.....	42
A.1 Funcions.....	42
A.2 Constant de Jacobi.....	42
A.3 Generació de punts inicials.....	43
A.4 ode78 .....	46
A.5 Newton-Rapshon.....	50
A.6 Càlcul de la secció.....	53
A.7 Càlcul dels Talls .....	54
A.8 Exploracions amb origen a Mart .....	57
A.9 Exploracions amb origen a la Terra .....	61
A.10 Anàlisi dels resultats .....	66
B- Pressupost.....	74

## 1 Introducció

De manera natural, a la Terra han arribat fragments de materials provinents de la part més externa del Sistema Solar. Per tal d'explicar aquest fenomen, l'estudi de la dinàmica de tot el Sistema Solar, tot i considerar exclusivament les atraccions gravitatòries dels cossos principals, és molt complexa. La idea és veure si utilitzant models més simples es pot explicar quin són els mecanismes naturals que expliquen el transport interplanetari.

La Mecànica Celeste és la disciplina que estudia el moviment dels cossos celestes degut a les seves atraccions gravitatòries i altres forces (com la radiació o la pressió solar). Des de mitjats del segle XX, la Astrodinàmica ha anat prenent més importància com a disciplina que estudia el moviment de cossos artificials i que incorpora elements de propulsió pròpia, navegació de satèl·lits en formació, etc. El meu treball es centra en el camp de la Mecànica Celeste, i voldré estudiar si hi ha mecanismes naturals degut exclusivament a forces gravitatòries que expliquin el transport de matèria de l'exterior del Sistema Solar a l'interior.

### 1.1 Antecedents

A l'article científic titulat *On the Mechanisms of Natural Transport in the Solar System*, realitzat per Ren Y, Masdemont J. J., Gómez G., Fantino E. de *Communications in Nonlinear Science and Numerical Simulations*, 17, 844, 2012. es van buscar indicis de transport natural dins el Sistema Solar, entenent per transport natural el fet que un cos pugui desplaçar-se d'un punt a un altre sense rebre cap força diferent de la gravitatòria dels cossos del Sistema Solar. El treball consistia en estudiar l'existència de connexions (òrbites) entre el sistema Sol-Mart i el sistema Sol-Terra. Els autors treballen amb un model simple, com és el RTBP (el qual tractarem amb detall en l'apartat 2.1) que conté uns punts d'equilibri, també anomenats Lagrangians, importants en la dinàmica com es veurà més endavant. El seu objectiu tracta de trobar connexions entre el punt d'equilibri  $L_1$  del sistema Sol-Mart i el punt d'equilibri  $L_2$  del sistema Sol-Terra (òrbites que sortint d'un entorn d'un dels punts d'equilibri anessin a parar a un entorn del segon), les quals no es van trobar. Prenent la idea, buscarem indicis de l'existència de transport natural usant altres punts d'equilibri dels dos sistemes. La motivació per realitzar aquest treball es deu a l'interès que sento per complementar els meus coneixements adquirits durant els estudis d'enginyeria en matemàtiques i en l'ús d'eines informàtiques per a l'estudi de models. Els càlculs numèrics s'han dut a terme usant el programa Matlab, la qual cosa m'ha permès poder

profunditzar els coneixements de programació en l'entorn Matlab que em podran ser útils de cara al món professional.

## 1.2 Objecte

Estudiarem el transport entre Mart i la Terra. Considerarem dos models per separat, que faran referència a el Sol, el planeta (Mart o la Terra) i una partícula; en el qual la partícula es mou sota l'atracció gravitatòria dels dos cossos celestes. L'objectiu és estudiar el transport natural dins cada model des del planeta a una secció fixada comuna als dos models, i després connectar els dos models. En primer lloc fixarem una secció intermèdia entre la Terra i Mart i buscarem si hi ha condicions inicials respecte el model Sol-Mart que ens permetin arribar a aquesta secció intermèdia. En segon lloc, respecte el model Sol-Terra, repetirem la simulació anterior però en sentit invers: sortint des del voltant d'un dels punts de Lagrange (coneguts) i considerarem el temps enrere, explorarem si hi ha condicions inicials que arriben a la mateixa secció intermèdia, per tal d'enllaçar amb les trajectòries provinents de Mart. Finalment, respecte el model Sol-Terra, estudiarem si hi ha connexions des del punt de Lagrange provinent de Mart amb el punt de Lagrange provinent de la Terra.

## 1.3 Especificacions i abast

Cal remarcar el fet que en aquest projecte d'investigació el limitem a buscar indicis de l'existència del transport natural, de manera que si el resultat fos positiu es podria realitzar un estudi més detallat de l'existència d'aquest.

Per a representar la dinàmica hem utilitzat el model de tres cossos restringits i hem desenvolupat diversos programes a través de Matlab que s'exposaran en el següents apartats del projecte. Simularem numèricament diverses condicions inicials al voltant dels punts de Lagrange per trobar quins paràmetres ens permetran arribar a la Terra. Per tal de realitzar l'estudi farem diferents simplificacions: considerarem només el Sol, la Terra i Mart, excloent els asteroides i la radiació solar; tindrem en compte les lleis de Newton i no considerarem les lleis relativistes. Aquest mateix estudi es pot realitzar considerant dos planetes qualsevols consecutius del Sistema Solar.

## 2 Model matemàtic

En aquest capítol analitzarem el model matemàtic amb el que estudiarem la possibilitat de transport natural en el Sistema Solar. Per transport natural entenem el fet que un cos pugui desplaçar-se d'un punt del sistema a un altre sense rebre cap força diferent de la gravitacional; és a dir que de forma natural, el cos es mou d'un lloc a un altre del sistema (en aquest cas el sistema Solar) gràcies a les forces gravitacionals i els moviments dels planetes que l'envolten.

Prendrem com a model bàsic el problema restringit a tres cossos, del qual comentarem també alguna particularitat que ens permetrà realitzar l'estudi.

### 2.1 Problema restringit a tres cossos

En anglès *restricted three body problem* que ens genera les sigles RTBP, correspon a un model de sistema on només considerarem el Sol, un planeta i un cos petit. Els dos cossos principals, també anomenats primaris, degut a les forces mútues d'atracció gravitatòria es mouen en òrbites circulars o el·líptiques respecte el centre de masses del conjunt. El tercer cos, de massa menyspreable respecte els primaris (un asteroide o un satèl·lit) no afecta al moviment dels dos primers, però el seu moviment ve determinat per les forces d'atracció que els primaris exerceixen sobre ell. El RTBP es centra en l'estudi del moviment d'aquest tercer cos.

Es realitza un canvi d'unitats de manera que la distància entre el Sol i el planeta és la unitat i el període, és a dir el temps que tarda el planeta en fer una volta al voltant del Sol, és dues vegades el nombre  $\pi$ . Considerem els dos primaris com a esferes perfectes i per tant el potencial gravitatori es pot concentrar en un sol punt, per tant podrem considerar que tenen masses puntuals.

A partir d'aquest canvi d'unitats aconseguim que la constant de gravitació per la suma de les masses del Sol i el planeta sigui unitària.

$$G(m_p + M_S) = 1 \quad (2.1.1)$$

D'aquesta manera el planeta tindrà massa  $\mu$ , el Sol  $(1 - \mu)$  i el cos petit es considera que té una massa negligible (el seu moviment ve definit únicament per a les forces

gravitacionals exercides per el Sol i el planeta). Per tant ens queda el problema en funció d'aquest paràmetre, que agafa el valor de:

$$\mu = \frac{m_p}{m_p + M_S} \quad (2.1.2)$$

En el nostre estudi explorem dos sistemes:

El sistema Sol-Terra on  $\mu = 3.00652690848 \cdot 10^{-6}$ .

El sistema Sol-Mart on  $\mu = 3.22775159768 \cdot 10^{-7}$ .

Prendrem un model RTBP en què que aquests tres cossos orbiten en el mateix pla, i que el planeta i el Sol tenen òrbites circulars respecte el seu centre de masses del sistema.

El RTBP és un sistema Hamiltonià, un tipus de sistemes dinàmics que formalitzen la descripció de problemes mecànics, encara que es poden utilitzar en altres contextos. Aquests sistemes són conservatius, hi ha una quantitat, anomenada funció de Jacobi (explicat amb més detall en l'apartat 2.3) que és constant sobre les solucions.

Aquest sistema s'acostuma a estudiar en un sistema en rotació amb la mateixa velocitat angular que els primaris, on aquests estan fixos sobre la recta que uneix els dos primaris, que correspondrà a l'eix de les  $x$ . Aquest fet ens fa aparèixer el terme de Coriolis en les equacions de govern, de manera que aquestes són:

$$\ddot{x} - 2\dot{y} = \frac{\partial \Omega}{\partial x} \quad (2.1.3)$$

$$\ddot{y} + 2\dot{x} = \frac{\partial \Omega}{\partial y} \quad (2.1.4)$$

On la funció Omega és la funció que conté el potencial gravitatori:

$$\Omega(x, y) = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu) \quad (2.1.5)$$

$$r_1^2 = (x - \mu)^2 + y^2 \quad (2.1.6)$$

$$r_2^2 = (x - \mu + 1)^2 + y^2 \quad (2.1.7)$$

De forma visual descrivim el RTBP amb la figura 1.

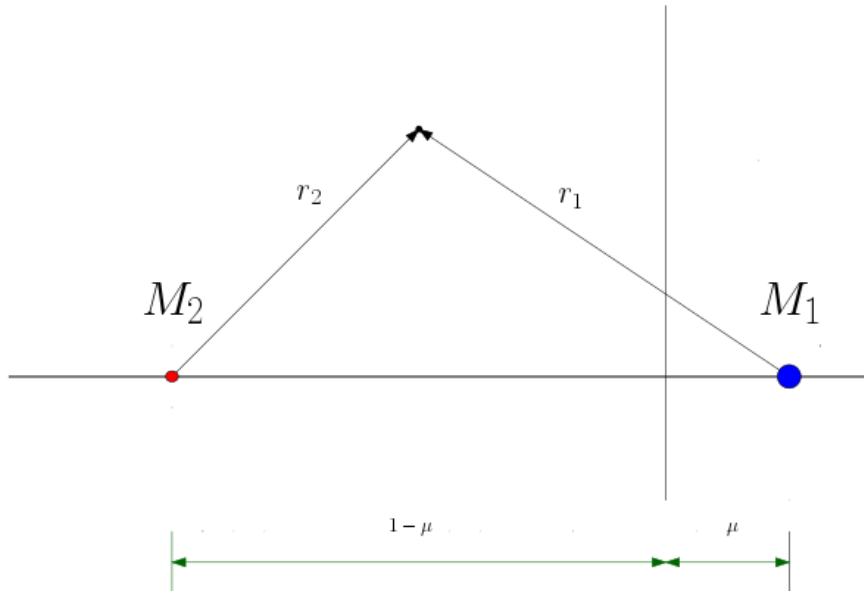


Figura 1: Representació del model restringit a tres cossos, on  $M_1$  correspon a la massa del Sol,  $M_2$  a la massa del planeta i el punt representa el tercer cos de massa menyspreable.

Re anomenem els termes:

$$x = x_1, \quad y = x_2, \quad x' = x_3, \quad y' = x_4$$

Desenvolupem les equacions:

$$\frac{d(\Omega)}{d(x_1)} = \frac{-\mu(x_1 - \mu + 1)}{(x_1^2 - 2(\mu - 1)x_1 + x_2^2 + (\mu - 1)^2)^{\frac{3}{2}}} + \frac{(\mu - 1)(x_1 - \mu)}{(x_1^2 - 2\mu x_1 + x_2^2 + \mu^2)^{\frac{3}{2}}} + x_1$$



$$\frac{d(\Omega)}{d(x)} = x'_3 - 2x_4$$

$$\frac{d(\Omega)}{d(x_2)} = \frac{-\mu x_2}{(x_2^2 + (x_1 - \mu + 1)^2)^{3/2}} + \frac{(\mu - 1)x_2}{(x_2^2 + (x_1 - \mu)^2)^{3/2}} + x_2$$

$$\frac{d(\Omega)}{d(y)} = x'_4 + 2x_3$$

D'on obtenim el sistema següent:

$$\left\{ \begin{array}{l} x'_1 = x_3 \\ x'_2 = x_4 \\ x'_3 = 2x_4 - \frac{\mu(x_1 - \mu + 1)}{(x_1^2 - 2(\mu - 1)x_1 + x_2^2 + (\mu - 1)^2)^{3/2}} + \frac{(\mu - 1)(x_1 - \mu)}{(x_1^2 - 2\mu x_1 + x_2^2 + \mu^2)^{3/2}} + x_1 \\ x'_4 = -2x_3 - \frac{\mu x_2}{(x_2^2 + (x_1 - \mu + 1)^2)^{3/2}} + \frac{(\mu - 1)x_2}{(x_2^2 + (x_1 - \mu)^2)^{3/2}} + x_2 \end{array} \right. \quad (2.1.8)$$

Observem que es tracta d'un sistema d'equacions diferencials de primer ordre que podem expressar tal com veiem a l'equació (2.1.9).

$$Z = (x, y, \dot{x}, \dot{y}) \quad (2.1.9)$$

$$\dot{Z} = F(Z)$$

D'aquesta manera aconseguim un sistema autònom ja que no depèn explícitament del temps. Per tant es poden donar punts d'equilibri, punts on la seva velocitat i acceleració seran nuls. Aquests punts d'equilibri corresponen a òrbites circulars en el sistema inercial original.

## 2.2 Punts d'equilibri

Independentment del valor del paràmetre  $\mu$ , el RTBP té cinc punts d'equilibri (anomenats també punts de Lagrange), tres dels quals són col·lineals amb els dos primaris (estan sobre la mateixa recta), i que es denoten per  $L_i$ ,  $i=1,2,3$ , i els altres dos

triangulars (formen un triangle equilàter amb els primaris). Conceptualment els punts de Lagrange són les solucions estacionàries del RTBP. Veure la figura 2.

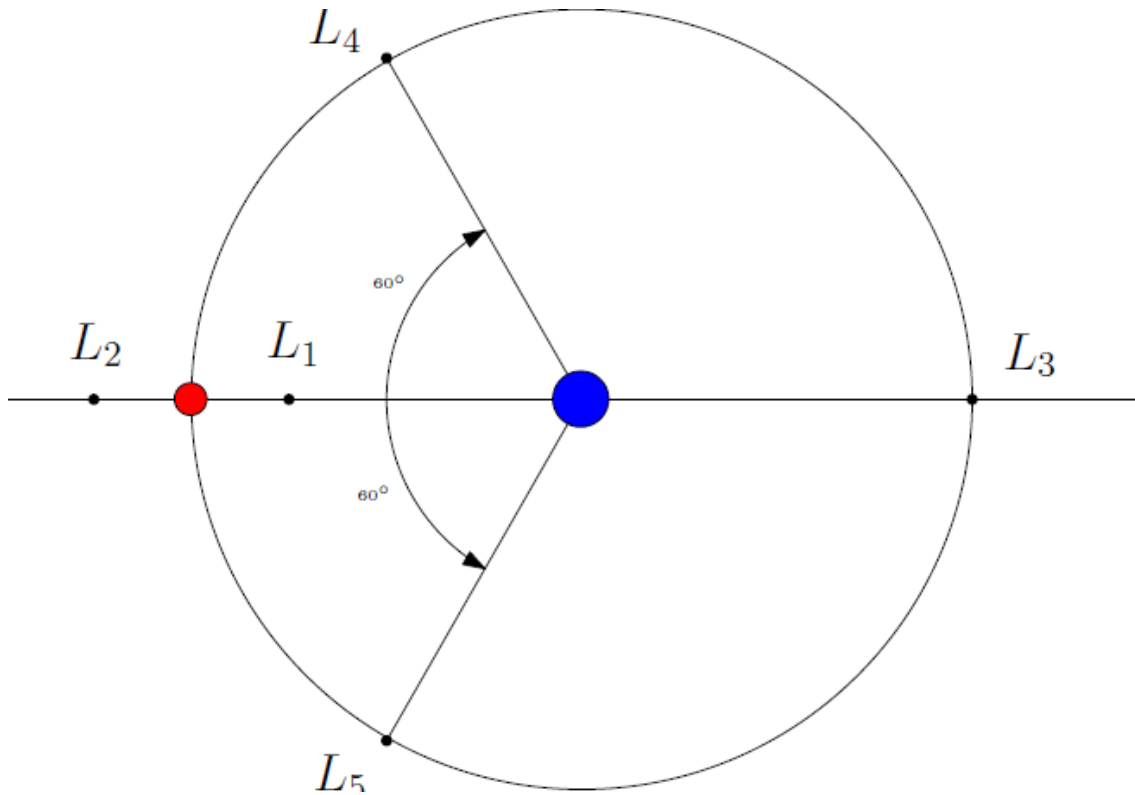


Figura 2: Representació dels punts d'equilibri (punts de Lagrange) en el model RTBP.

L'estudi de la dinàmica del sistema consisteix en entendre el tipus de comportaments que es poden obtenir, això és, quin tipus de solucions (òrbites) té el sistema. La dinàmica global és molt complexa, però podem estudiar la dinàmica al voltant dels punts d'equilibri que són els que ens interessin. Observem que es tracta d'un sistema d'equacions diferencials de primer ordre, tal com hem vist en l'equació (2.1.9).

Partim d'un punt d'equilibri  $L_i$  i li apliquem petits desplaçaments  $\Delta z$  per a veure com es comporta el sistema:

$$Z = L_i + \Delta z \rightarrow \dot{Z} = F(L_i + \Delta z)$$

Apliquem Taylor de primer ordre per tal de calcular-ne l'aproximació:

$$\dot{Z} = F(L_i) + DF(L_i)\Delta z + O_2 \rightarrow \dot{\Delta z} = DF(L_i)\Delta z \quad (2.2.1)$$

Menyspreant els termes d'ordre dos, el comportament de solucions molt properes als punts d'equilibri vindrà donat per les solucions de l'equació (2.2.1). Aquest sistema és lineal i és conegut que les solucions depenen dels valors propis de la matriu  $DF(L_i)$ : segons com siguin aquests tindrem solucions trigonomètriques i solucions exponencials.

### 2.2.1 Punts d'equilibri triangulars

Corresponen als punts  $L_4$  i  $L_5$ , que com veiem a la figura 2 ens generen un triangle equilàter amb el planeta i el Sol.

En aquest cas els valors propis de la matriu  $DF(L_i)$  són complexos purs per a valors del paràmetre  $\mu$  petits. Això vol dir que els punts són linealment estables (recordem que hem menyspreat termes d'ordre 2) i que en primer ordre les solucions són combinacions de funcions trigonomètriques i per tant tindrem solucions quasi-periòdiques (en particular, seran òrbites que romandran donant voltes al voltant del punt durant molt temps, si no per sempre).

Com a comentari, en el cas del sistema Sol-Júpiter és conegut que en els punts d'equilibri  $L_4$  i  $L_5$  trobem els anomenats asteroides Troians (aquest nom ve del fet que se'ls va anomenar en honor als defensors de Troia) i els asteroides Grecs (que van ser anomenats així en honor als guerrers grecs), que segueixen una trajectòria que en general no escapa l'entorn del respectiu punt d'equilibri. Veure la figura 3.

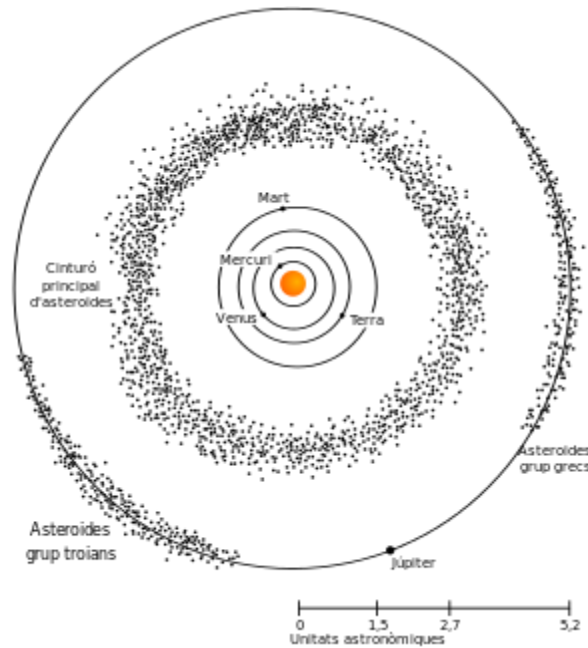


Figura 3: Representació dels asteroides Grecs i Troians. FONT: [https://ca.wikipedia.org/wiki/Asteroides\\_troians](https://ca.wikipedia.org/wiki/Asteroides_troians).

### 2.2.2 Punts d'equilibri col·lineals

La posició dels punts d'equilibri en funció del paràmetre de masses és coneguda (veure per exemple [2]). Sabent que la seva coordenada en l'eix de les ordenades és zero, la seva posició en l'eix de les abscisses ve determinat per les equacions (2.2.2.1), recordem la figura 2 on es mostraven aquests punts d'equilibri.

$$\begin{aligned}
 x_1 &= -1 + \left(\frac{\mu}{3}\right)^{\frac{1}{3}} - \frac{1}{3}\left(\frac{\mu}{3}\right)^{\frac{2}{3}} + \frac{26}{9}\left(\frac{\mu}{3}\right) + O(\mu^{\frac{4}{3}}) \\
 x_2 &= -1 - \left(\frac{\mu}{3}\right)^{\frac{1}{3}} - \frac{1}{3}\left(\frac{\mu}{3}\right)^{\frac{2}{3}} + \frac{28}{9}\left(\frac{\mu}{3}\right) + O(\mu^{\frac{4}{3}}) \\
 x_3 &= 1 + \frac{5}{12}\mu + O(\mu^3)
 \end{aligned}
 \tag{2.2.2.1}$$

Són els punts que ens interessen més per realitzar el projecte, correspon als punts  $L_1, L_2$  i  $L_3$ . Per tal d'analitzar-ne la seva estabilitat d'una forma experimental vam llançar diversos trajectòries amb aquest origen per a estudiar-ne el comportament.

En aquest cas els valors propis de la matriu  $DF(L_i)$  són del tipus:

$$\lambda, -\lambda, +i\omega, -i\omega$$

De manera que les solucions de l'equació (2.1.9) contenen exponencials positives i negatives, així l'aproximació d'ordre u de la solució de l'equació és de la forma (per a cada coordenada):

$$Ae^{\lambda t} + Be^{-\lambda t} + C \cos(\omega t) + D \sin(\omega t) \quad (2.2.2.2)$$

on la part exponencial ens generà la part hiperbòlica (d'escapament en temps més i menys infinit) i la trigonomètrica la part central (generarà moviment periòdic o quasi periòdic). Observem vàries particularitats:

- Si  $A$  i  $B$  agafessin el valor de zero obtindríem moviments acotats (és a dir que no tendeixen a infinit).
- Si  $A$  té valor i  $B, C$  i  $D$  no, la partícula s'escapa del punt d'equilibri en un temps futur, i prové del punt d'equilibri en temps passat.
- Si  $B$  té valor i  $A, C$  i  $D$  no, la partícula arribarà (s'hi acostarà) al punt d'equilibri en temps futur.

Aquests punts col·lineals són inestables ja que és molt fàcil que  $A, B, C$  i  $D$  agafin un valor i ens provoquin que la partícula marxi.

Aquests punts d'equilibri generen òrbites periòdiques al seu voltant. És conegut que aquestes òrbites periòdiques són inestable en el mateix sentit, és a dir, hi ha un conjunt d'òrbites que s'escapen en temps futur i que en el passat estaven molt a prop de la òrbita (s'anomena varietat inestable) i un altre conjunt que s'acosten a l'òrbita en temps futur (s'anomena varietat estable). Tindrem doncs conjunts d'òrbites d'entrada i altres de sortida en un entorn del punt d'equilibri.

### 2.3 Energia associada

El nostre model matemàtic, és conservatiu: això vol dir que hi ha una quantitat que es conserva al llarg de les trajectòries, això és, tota solució del problema (òrbita) té una

quantitat associada que és la mateixa a tots els punts. S'anomena constant de Jacobi i està definida a través de la següent expressió:

$$\dot{x}^2 + \dot{y}^2 = 2\Omega(x, y) - C_J \rightarrow C_J = 2\Omega(x, y) - \dot{x}^2 - \dot{y}^2 \quad (2.3.1)$$

Recordem que el terme  $\Omega$  l'hem descrit en l'equació 2.1.5.

A la constant de Jacobi li diem energia ja que així és com s'anomena la quantitat  $H = -C_J/2$  que és constant.

Els punts d'equilibri doncs tindran una energia associada que vindrà determinada per les següents expressions (veure [2]):

$$\begin{aligned} C_{J_1} &= 3 + 9\left(\frac{\mu}{3}\right)^{\frac{2}{3}} - 7\left(\frac{\mu}{3}\right) + O(\mu^{\frac{4}{3}}) \\ C_{J_2} &= 3 + 9\left(\frac{\mu}{3}\right)^{\frac{2}{3}} - 11\left(\frac{\mu}{3}\right) + O(\mu^{\frac{4}{3}}) \\ C_{J_3} &= 3 + 2\mu - \frac{49}{48}\mu^2 + O(\mu^3) \end{aligned} \quad (2.3.2)$$

Gràcies a que el nostre model és un sistema conservatiu aquesta energia es conserva al llarg del recorregut. D'aquesta manera podem estudiar el nostre sistema per a diferents capes d'energia.

Observem una característica del model important. De la relació de la constant de Jacobi, si aïllem el mòdul de la velocitat, com que ha de ser una quantitat positiva, tenim la restricció representada en l'equació (2.3.3).

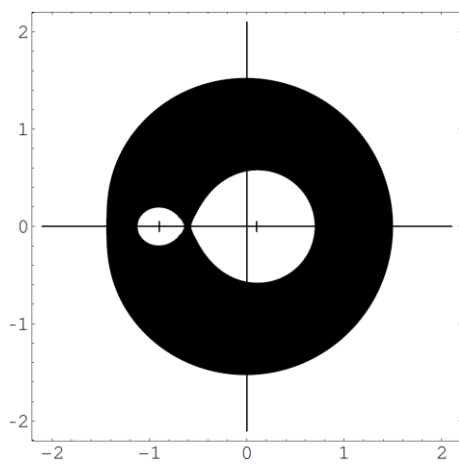
$$|\vec{V}| = \sqrt{2\Omega(x, y) - C_J} \rightarrow 2\Omega(x, y) - C_J > 0 \quad (2.3.3)$$

Això vol dir que, fixat un nivell d'energia o  $C_J$ , no totes les regions del pla  $(x, y)$  són admissibles. Estudiem ara les zones on ens podem moure, delimitades per quan el

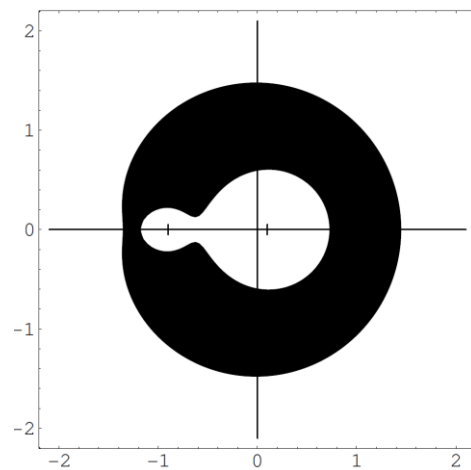
moviment del tercer cos és zero, és a dir que els seus valors de velocitat són nuls; que correspon a la següent expressió:

$$C_J = 2\Omega(x, y) \quad (2.3.4)$$

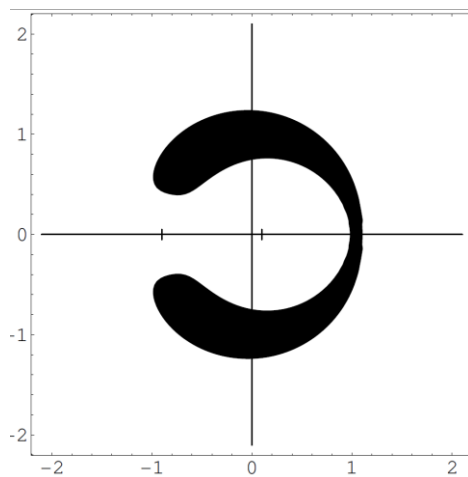
Veiem doncs que per a cada constant de Jacobi tindrem un rang de valors possibles de posició, i zones prohibides. D'aquesta manera obtenim la figura 4.



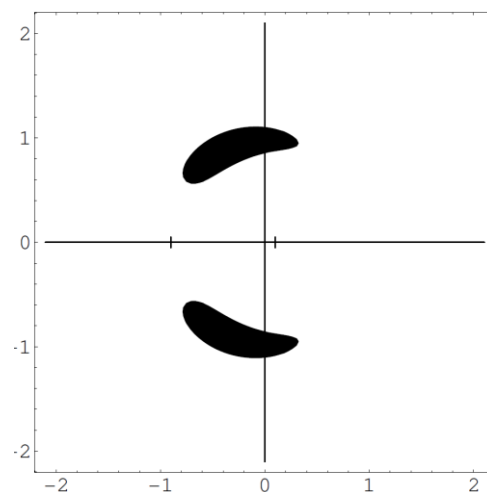
$$C_J > C_{J1}$$



$$C_{J2} < C_J < C_{J1}$$



$$C_{J3} < C_J < C_{J2}$$



$$3 < C_J < C_{J3}$$

Figura 4: Representació de la regió que no es podrà recórrer a partir d'una constant de Jacobi. FONT:[2].

L'àrea ombrejada de la figura 4 correspon a la regió que no podrà recórrer el cos amb la constant de Jacobi determinada. En primer lloc com està indicat tenim el cas en que la nostra energia és superior a la del primer punt d'equilibri, veiem que en aquest cas no podríem accedir del planeta al Sol. Si agafem un valor d'energia entre els valors d'energia del primer punt d'equilibri i el segon, veiem que es crea un canal que ens permet tenir trajectòries que puguin anar del planeta al Sol. En la tercera imatge agafem un valor de la constant de Jacobi entre l'energia del segon punt d'equilibri i el tercer de manera que aconseguim tenir una zona restringida inferior. Per últim estudiem un valor de la constant de Jacobi entre tres i la constant de Jacobi del tercer punt d'equilibri i veiem que tenim una regió inaccessible encara més petita que l'anterior.

Cal dir, que els valors de la constant de Jacobi en els punts d'equilibri coincideixen amb els moments en que les zones de moviment admissible es van obrint.

En el nostre cas doncs, haurem de treballar amb valors de la constant de Jacobi inferiors al del punt  $L_1$  (de manera que la regió de moviment ja s'ha obert), però a prop de  $C_J(L_1)$  per aprofitar-nos de la dinàmica de l'entorn del punt.



### 3 Metodologia tècnica

En aquest capítol s'exposen els conceptes metodològics aplicats en aquest treball.

#### 3.1 Mètode d'integració numèrica Runge-Kutta-Fehlberg

En general, és difícil trobar solucions explícites (en funció de la variable independent) d'un sistema d'equacions diferencials. Aquest és el cas del RTBP i per tal d'estudiar el comportament de les solucions, caldrà trobar-les numèricament.

Els mètodes *Runge-Kutta* són un conjunt de mètodes genèrics iteratius, explícits o implícits, de resolució numèrica d'equacions diferencials; que van ser desenvolupats l'any 1900 per els matemàtics *C. Runge* i *M. W. Kutta*. Posteriorment *Erwin Fehlberg* basant-se en el model *Runge-Kutta* va millorar l'algoritme, realitzant un càlcul addicional que permet estimar i controlar l'error de la solució així com determinar automàticament la longitud dels passos amb els que el mètode avança per desenvolupar el càlcul de la solució. Obtenint un mètode eficient per a resoldre problemes d'integració numèrica d'equacions diferencials ordinàries.

En primer lloc, per tal de practicar i entendre millor el mètode numèric, vam desenvolupar a través del software *Matlab* els mètodes numèrics *Runge-Kutta 3*, *Runge-Kutta 4* i *Runge-Kutta-Fehlberg 34*.

Inicialment havíem escollit treballar amb el mètode numèric *Runge Kutta Fehlberg* d'ordre 7 i 8 per a tal de resoldre el RTBP. Els nostres càlculs els hem realitzat a través de la plataforma *Matlab* de manera que vam aplicar una funció lliure ja desenvolupada que aplicava aquest mètode (*ode78*, veure l'apartat A.4 de l'annex), aquest mètode ens donava una elevada precisió però té el problema que la funció aplicada no podia integrar amb temps negatiu, cosa que necessitem en una part del projecte. La funció *ode78* gairebé no l'hem utilitzat, degut a aquest problema.

Finalment doncs, hem acabat treballant amb el mètode numèric *Runge Kutta Fehlberg* d'ordre 4 i 5 per a resoldre el RTBP. Ja que *Matlab* conté una funció robusta (*ode45*) que resol aplicant aquest mètode equacions diferencials ordinàries. Aquest mètode per a treballar a temps més llargs és menys correcte que l'anterior ja que la precisió amb la que treballa és més baixa. De totes maneres com nosaltres farem exploracions a curt termini ja ens serveix.

### 3.2 Mètode de Newton-Raphson

El mètode de Newton-Raphson és un algoritme eficient per a trobar aproximacions d'arrels en una funció real. També pot ser aplicat per a trobar el màxim o el mínim d'una funció, trobant els zeros de la primera derivada.

Va ser descrit per Isaac Newton al 1669, però només l'aplicava en polinomis, i no considerava les aproximacions successives, sinó que calculava una seqüència de polinomis per a arribar a la aproximació de l'arrel. Newton només contemplava el mètode com a purament algebraic, i no li troba una connexió amb el càlcul. Va ser Joseph Raphson qui va aplicar aquest mètode per a aproximar arrels, d'aquí el nom del mètode.

Utilitzarem aquest mètode per tal de trobar els talls en la secció desitjada, inicialment quan la coordenada  $x$  val zero i finalment en la secció  $x = \mu$ , a partir d'un punt anterior i un posterior a aquesta secció. Veure l'apartat A.5 de l'annex.

### 3.3 Càlcul dels punts inicials

Com hem vist en apartats anteriors (i tal com expliquem en l'apartat 4.1) els punts inicials els situarem prop de  $L_1$  de Mart i prop de  $L_3$  de la Terra. Experimentalment determinem la distància màxima a la que es poden situar els punts inicials d'aquests punts (veure l'apartat A.3.8 de l'annex), que en el cas de Mart són aproximadament 0,0016 *unitats* (recordem que cada sistema té el seu propi sistema d'unitats) i en el cas de la Terra 0,06 *UA*. D'aquesta manera queda determinat el rang de valors de posició que poden agafar els punts inicials. La posició doncs vindrà determinada per la expressió següent:

$$x = x_{Li} + \Delta x$$

$$y = \Delta y$$

A partir del valor de la constant de Jacobi i la posició, a través de l'equació (2.3.1) podem trobar el valor del mòdul de la velocitat de la partícula:

$$|\vec{V}| = \sqrt{2\Omega(x, y) - C_J}$$

L'angle el determinarem segons el nombre de velocitats que desitgem simular per a cada punt, de manera que tots els angles siguin equidistants. Així doncs l'angle vindrà donat per la expressió següent:

$$\alpha = \frac{360}{n} + k \cdot n$$

On  $n$  és el nombre de velocitats que desitgem per a cada punt i  $k$  és un valor enter que va de zero a  $n$ . D'aquesta manera els paràmetres de la velocitat els calcularem amb la expressió:

$$x' = |\vec{V}| \cdot \cos(\alpha)$$

$$y' = |\vec{V}| \cdot \sin(\alpha)$$

### 3.4 Conversió del sistema de referència

Tal i com hem descrit en l'apartat 2.1 cada un dels dos sistemes tindrà el seu propi sistema de coordenades, cosa que ens facilita els càlculs. Tot i això a la hora de contrastar els resultats entre un sistema i l'altre hem d'adaptar un dels dos sistemes. Per a resoldre aquest problema, adaptarem els resultats del sistema Sol-Mart en coordenades del sistema Sol-Terra, per a fer-ho haurem de seguir les expressions que s'expliquen a continuació.

En primer lloc haurem de calcular dos factors, el primer correspondrà a la relació entre les distàncies planeta-Sol. La distància entre el Sol i la Terra són  $149.6 \cdot 10^6 \text{ km}$  i la distància entre el Sol i Mart són  $227.9 \cdot 10^6 \text{ km}$ .

$$Conv = \frac{227.9 \cdot 10^6}{149.6 \cdot 10^6}$$

El segon terme farà referència en la conversió temporal. El període de translació de la Terra és  $365.256363004 \text{ dies}$  i el període de translació de Mart és  $686.971 \text{ dies}$ .

$$ConvT = \frac{686.971}{365.256363004}$$

D'aquesta manera la conversió temporal s'expressa com:

$$tempsC = ConvT \cdot temps$$

La conversió de la posició  $x$ :

$$xC = \mu_T - Conv \cdot (x - \mu_M)$$

La conversió de la posició  $y$ :

$$yC = Conv \cdot y$$

La conversió de la velocitat  $x'$ :

$$x'C = \mu_T - \frac{Conv}{ConvT} \cdot (x' - \mu_M)$$

La conversió de la velocitat  $y'$ :

$$y'C = \frac{Conv}{ConvT} \cdot y'$$

Veure l'apartat A.10.7 de l'annex.

## 4 Plantejament i desenvolupament

En aquest capítol s'explica com hem desenvolupat el projecte.

### 4.1 Plantejament

Per veure si trobem transport entre Mart i la Terra, ho farem en tres fases:

Fixarem una secció intermèdia entre la Terra i Mart, i en el model Sol-Mart, farem una exploració sortint del voltant del punt d'equilibri  $L_1$  per detectar si hi ha condicions inicials que arribin a aquesta secció intermèdia.

En el model Sol-Terra, repetirem la simulació anterior, però ara sortirem del voltant del punt d'equilibri  $L_3$  i (anant temps enrere) explorarem si hi ha condicions inicials que arriben a la mateixa secció que a l'apartat anterior, i poden empalmar amb les trajectòries provinents de Mart.

En el model Sol-Terra, estudiarem si hi ha connexions del punt d'equilibri  $L_3$  amb la Terra. Per això es farà una simulació numèrica llençant un cert nombre de condicions inicials al voltant de  $L_3$  i es buscarà per a quins paràmetres s'arriba a la Terra.

Es conegut que existeix transport natural en el sistema Sol-Terra entre  $L_3$  i la Terra. El plantejament queda representat per la figura 5.

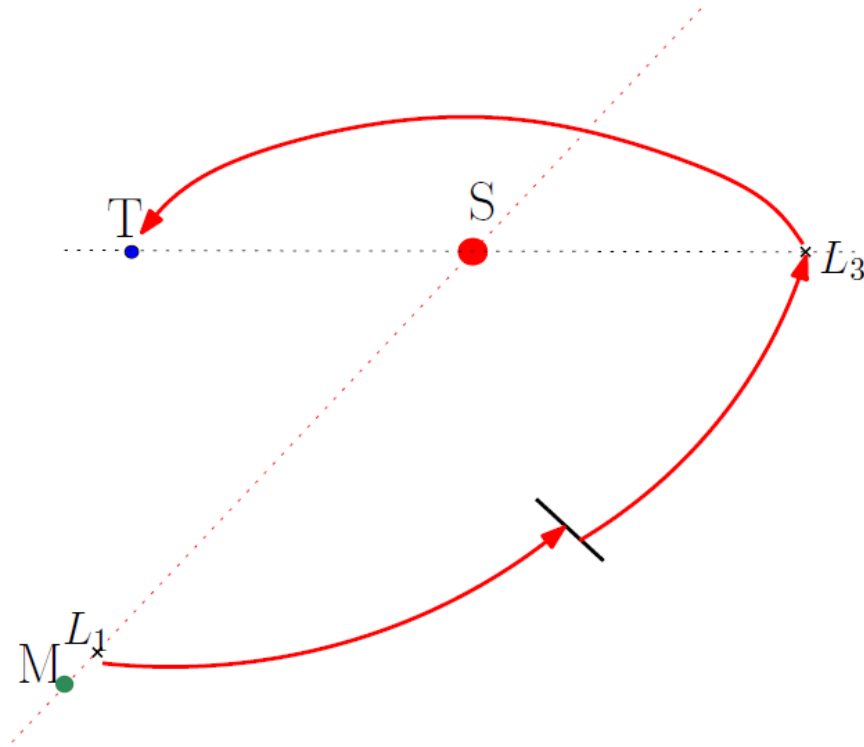


Figura 5: Representació qualitativa de transport entre Mart i la Terra passant per alguns punts d'equilibri de cada model.

#### 4.2 Càlcul de trajectòries d'òrbites

El primer programa que vam desenvolupar, amb l'objectiu de poder interpretar les trajectòries dels punts estudiats va ser el que anomenem per *Càlcul orbital* el qual vam adaptar per a cada sistema (veure els apartats A.8.1 i A.9.1 de l'annex). Per a desenvolupar-lo vam generar una funció la qual conté el sistema d'equacions de govern del sistema (veure l'apartat A.1 de l'annex) necessària per a executar les funcions *ode78* i *ode45* per a cada cas. D'aquesta manera aconseguim un programa que a partir d'un punt inicial ens calcula i representa visualment la trajectòria del tercer cos del RTBP en el sistema, amb una precisió de  $10^{-6}$  unitats, i un pas màxim d'integració de  $10^{-1}$  unitats, per un temps determinat.

Com a exemple podem representar la trajectòria del punt inicial  $x = 1, y = 1, x' = 1, y' = -1$  en el sistema Sol-Mart, per un temps màxim de cinc-cens unitats, tal com s'aprecia en la figura 6.

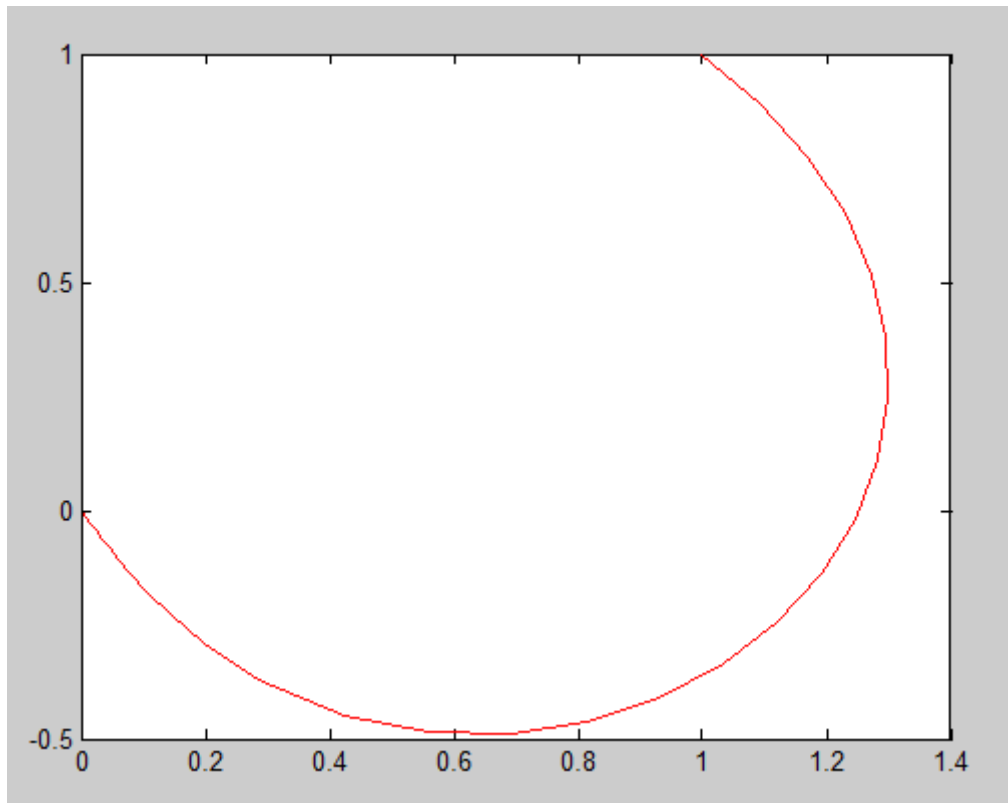


Figura 6: Representació de la trajectòria del punt  $x = 1, y = 1, x' = 1, y' = -1$  en el sistema Mart-Sol.

#### 4.2 Integració a secció

Amb l'objectiu de desenvolupar un algoritme que ens trobi els talls d'una trajectòria en una secció determinada vam desenvolupar la funció *TallsT* de la qual vam fer versions per a cada sistema (veure l'apartat A.7 de l'annex).

A partir d'un punt inicial, utilitzem la funció *ode45* amb la que obtenim un seguit de punts que són una aproximació de la corresponent trajectòria. A la que trobem dos punts on entre ells la trajectòria travessa la secció apliquem el mètode de Newton-Raphson per a determinar el valor de la trajectòria en la secció (veure apartat A.5 de l'annex). D'aquesta manera trobem els talls de la trajectòria del tercer cos del RTBP fins a un determinat temps, amb una precisió de  $10^{-10}$  unitats.

#### 4.3 Càlcul dels punts inicials

Com hem explicat, el nostre procés es basa en llançar un conjunt de condicions inicials properes als punts  $L_1$  de Mart i  $L_3$  de la Terra. Per tal de fer-ho vam desenvolupar les funcions *PI\_M* i *PI\_T* (veure els apartats A.3.1 i A.3.2 de l'annex), una per a cada

sistema. Aquestes funcions ens generen (basant-nos en l'explicat en l'apartat 3.3) una matriu de punts que es troben dins un rang de valors propers al punt d'equilibri corresponent, per entendre-ho de forma visual podem veure l'exemple de la figura 7, on el centre de la matriu correspon al punt d'equilibri.

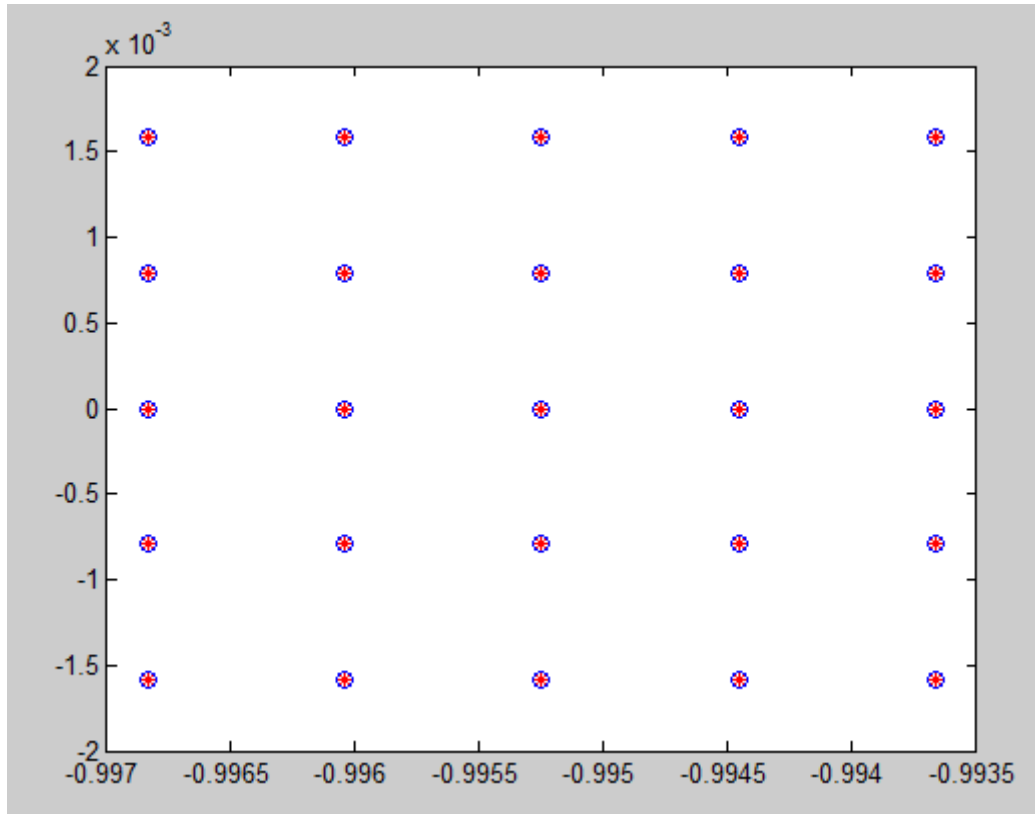


Figura 7: Representació d'una petita mostra de punts inicials.

A cada punt li assignarem diversos valors de velocitat, tal com expliquem en l'apartat 4.3.2.

#### 4.3.1 Càlcul de la posició dels punts

Per tal de realitzar el càlcul de les condicions inicials hem desenvolupar un seguit de funcions. En primer lloc vam situar els punts d'equilibri en el nostre sistema de coordenades, per a realitzar-ho vam desenvolupar les funcions  $L1.m$  i  $L3.m$  (veure els apartats A.3.4 i A.3.5 de l'annex). Un cop situats els punts centrals, determinem una distància per a generar la matriu quadrada tal com hem explicat en l'apartat 3.3.

A partir del punt Lagrangia i coneixent la distància vam generar la funció  $GeneracioM\_2$  (veure l'apartat A.3.3 de l'annex) la qual ens genera una matriu de



punts inicials tal com hem vist en la figura 7, de manera que en el centre dels punts de la matriu es trobarà el punt Lagrangià.

#### 4.3.2 Càlcul de la velocitat dels punts

Tal com hem explicat en l'apartat 3.3, a partir d'un valor de la constant de Jacobi determinem el valor del mòdul de la velocitat per a cada punt, i a partir del nombre de velocitats determinem el valor dels angles del vector velocitat. A partir d'aquest criteri vam desenvolupar la funció *GeneracioPI* (veure l'apartat A.3.7 de l'annex), la qual assigna els valors de les velocitats a les posicions dels punts inicials.

#### 4.4 Càlcul de trajectòries per a varis punts

Un cop desenvolupades les funcions que ens generen els punts inicials i ens determinen els talls en una secció vam desenvolupar un seguit de programes, per a cada sistema, que ens englobessin les dues funcions. Pel que fa al sistema Sol-Mart, el programa integrarà en temps positiu les condicions inicials i ens trobarà el tall de les seves trajectòries en la secció. En canvi en el sistema Sol-Terra el programa integrarà en temps negatiu, per tal d'arribar a la secció. D'aquesta manera creem trajectòries que vagin des de  $L_1$  de Mart fins a la secció i de la secció fins a  $L_3$  de la Terra.

Inicialment vam calcular els talls per una sola constant de Jacobi i en la secció  $x = 0$ . Aquests programes són els corresponents als apartats A.8.2 i A.9.2 de l'annex, on per un cert nombre de punts inicials, una constant de Jacobi, un temps màxim per a cada trajectòria, una precisió de  $10^{-6}$  unitats i un pas màxim d'integració de  $10^{-1}$  unitats ens calcula els talls en la secció per a cada condició inicial.

Un cop desenvolupats aquests dos programes vam elaborar una versió d'aquests que treballessin per un rang de constant de Jacobi enlloc de per un sol valor, també sobre la secció  $x = 0$ . Aquests programes són els corresponents als apartats A.8.3 i A.9.3 de l'annex. En aquest cas generàvem valors de la constant de Jacobi dins un rang determinat (tal com hem comentat en l'apartat 2.3) en funció del nombre de constants que volguéssim explorar.

Finalment per tal de poder contrastar un sistema amb l'altre vam determinar que la secció més correcte amb la que treballar correspon a on es troba el Sol, és a dir  $x = \mu$ . Seguint aquest criteri vam modificar els programes mencionats en aquest apartat per

tal de complir aquest requisit. D'aquesta manera vam obtenir els programes dels apartats A.8.4 i A.9.4 de l'annex, els quals tenen la mateixa estructura que els anteriors però busquen els talls sobre la secció  $x = \mu$ . Aquests programes són els que hem utilitzat per tal de realitzar les exploracions de cada sistema.

## 4.5 Contrast de dades

### 4.5.1 Conversió d'unitats

Un cop executades les exploracions per a cada sistema les hem de contrastar, però ens trobem el problema de que cada una es troba en el seu propi sistema. Basant-nos en l'explicat en l'apartat 3.4, vam desenvolupar la funció *ConversioM* (veure l'apartat A.10.7 de l'annex). D'aquesta manera aconseguim transformar els resultats obtinguts en l'exploració del sistema Sol-Mart en coordenades del sistema Sol-Terra.

### 4.5.2 Cerca de coincidències

Finalment vam desenvolupar un seguit de programes per tal de contrastar les dades. Per tal de valorar en un inici si anàvem ben encaminats amb la busca d'indícis, vam elaborar un programa el qual amb una precisió de  $10^{-6} UA$  ens analitzi si hi ha punts coincidents entre les dues exploracions en la secció (veure l'apartat A.10.1 de l'annex).

Seguidament vam elaborar un segon programa el qual ens analitza si hi havia coincidències de posició amb una precisió de  $10^{-6} UA$  i si aquests punts coincidien en velocitat per una precisió de  $10^{-3} unitats$  (veure apartat A.10.2 de l'annex).

Finalment vam desenvolupar un programa que ens analitza els resultats de les dues exploracions en busca de coincidències de posicions per una precisió de  $10^{-6} UA$  i ens gràfiques les diferències de velocitats per a cada coincidència de posició (veure l'apartat A.10.3 de l'annex).

## 5 Resultats

En aquest capítol s'exposa un anàlisi visual del tipus de trajectòries amb les que treballem i el resultat de les exploracions realitzades en busca de talls en la secció  $x = \mu$  per els dos sistemes estudiats i el resultat de contrastar-les.

### 5.1 Anàlisi visual de les òrbites

Realitzarem de forma visual, un breu anàlisi de les òrbites que es podran donar en les nostres exploracions, a partir de petites mostres de punts inicials. Realitzarem gràfics on es compararà la coordenada  $X$  respecte la  $Y$  per a poder analitzar la trajectòria.

Executarem els programes dels apartats A.8.3 o A.9.3 de l'annex depenent del sistema, per tal d'obtenir els talls dels punts inicials en la secció  $x = 0$ . I posteriorment analitzarem la trajectòria d'alguna de les condicions inicials que ens generen talls a partir dels programes de l'apartat A.8.1 o A.9.1 de l'annex, també depenent del sistema.

Per tal d'analitzar els talls gràficament utilitzarem la secció de Pointcaré. Com hem explicat en l'apartat 2.3 podem expressar els punts segons l'expressió:

$$C_J = 2\Omega(x, y) - \dot{x}^2 - \dot{y}^2$$

Així doncs un punt queda representat per a quatre paràmetres. En el nostre cas la constant de Jacobi està fixada i el valor de la coordenada  $x$  també, per tant els punts tenen dos graus de llibertat. D'aquesta manera justifiquem que podem representar els punts dels talls en la secció en funció dos paràmetres:  $Y$  i  $Y'$ .

#### 5.1.1 Òrbites amb origen $L_3$ de la Terra

Si executem una exploració de la Terra per una matriu de punts inicials  $5 \times 5$  (situats en el  $L_3$ ) explorant cinc velocitats per a cada punt inicial, explorant tres constants de Jacobi i un temps de dos-cents unitats; obtenim la figura 8 comparant les  $Y$  amb les  $Y'$ :

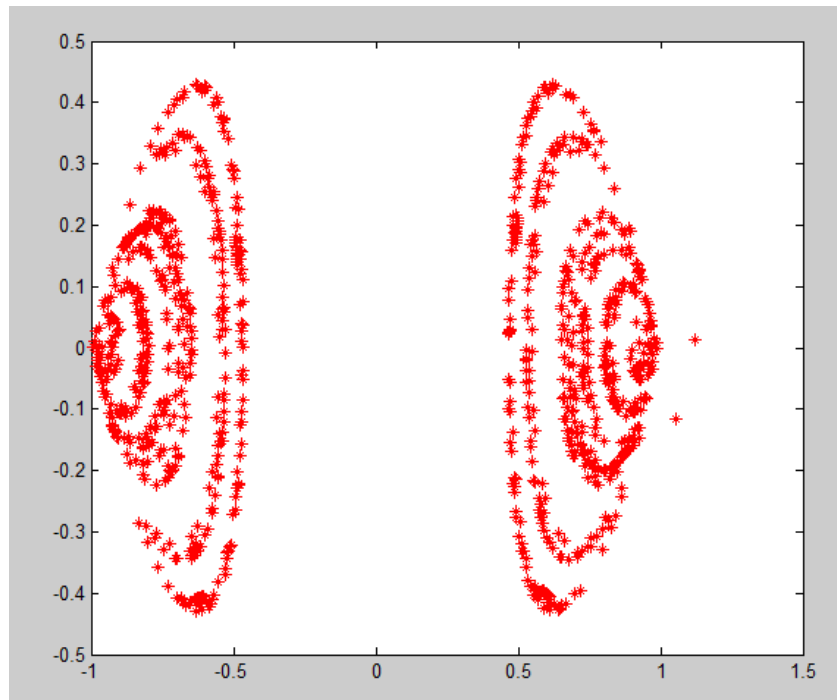


Figura 8: Representació de les posicions de  $Y$  contra la velocitat  $Y'$  per una petita mostra de punts amb origen  $L_3$  de la Terra.

Analitzem un parell de punts inicials qualsevols representatiu de l'exploració com per exemple el punt:

$$temps = 0, \quad X = 0.940001252719545, Y = -0.0600000000000000,$$

$$X' = 0.0317227919384568, Y' = 0.0976327145022242, C_J = 3.00000601304459$$

I el punt:

$$temps = 0, X = 0.940001252719545, Y = -0.0600000000000000,$$

$$X' = 0.102657111148928, Y' = 0, C_J = 3.00000601304459$$

A partir del programa *CalculOrbital\_1T.m* (veure l'apartat A.9.1) estudiem les seves òrbites per un temps de cinc-cents unitats, d'on obtenim la figura 9. Tal com podem veure aquest punt inicial ens genera talls en la secció per a valors de  $Y$  positius i per a valors negatius, generant-nos la simetria que podem observar en la figura 8.

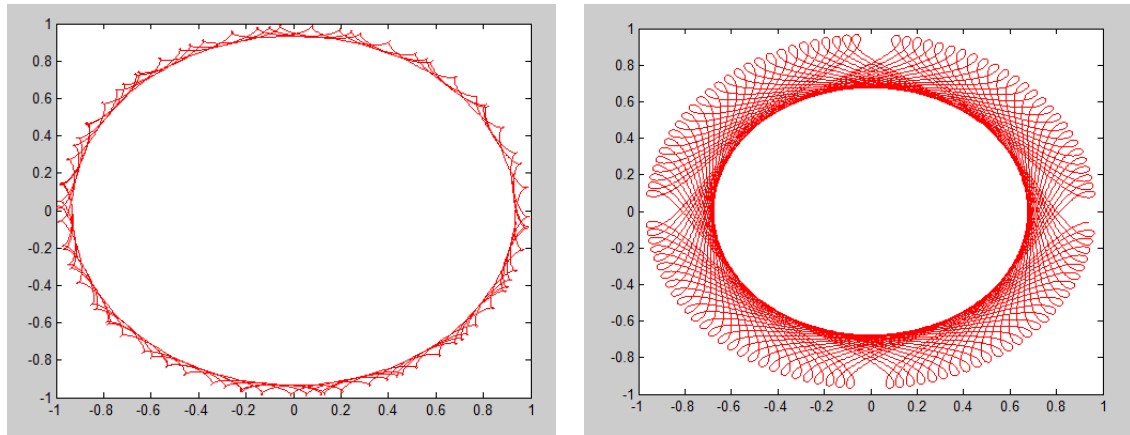


Figura 9: Representació de dues trajectòries en el sistema Terra-Sol.

Finalment analitzem els punts més allunyats de la figura 8, ja que ens criden l'atenció perquè surten de la simetria general.

Veiem que els dos punts tenen el mateix origen, per tant analitzem la trajectòria del punt:

$$temps = 0, X = 0.940001252719545, Y = 0.0300000000000000,$$

$$X' = 0.0325229843815117, Y' = 0.100095453610786, C_J = 3.00000601304459$$

Tornem a executar el programa *CalculOrbital\_1T.m* i obtenim la figura 10.

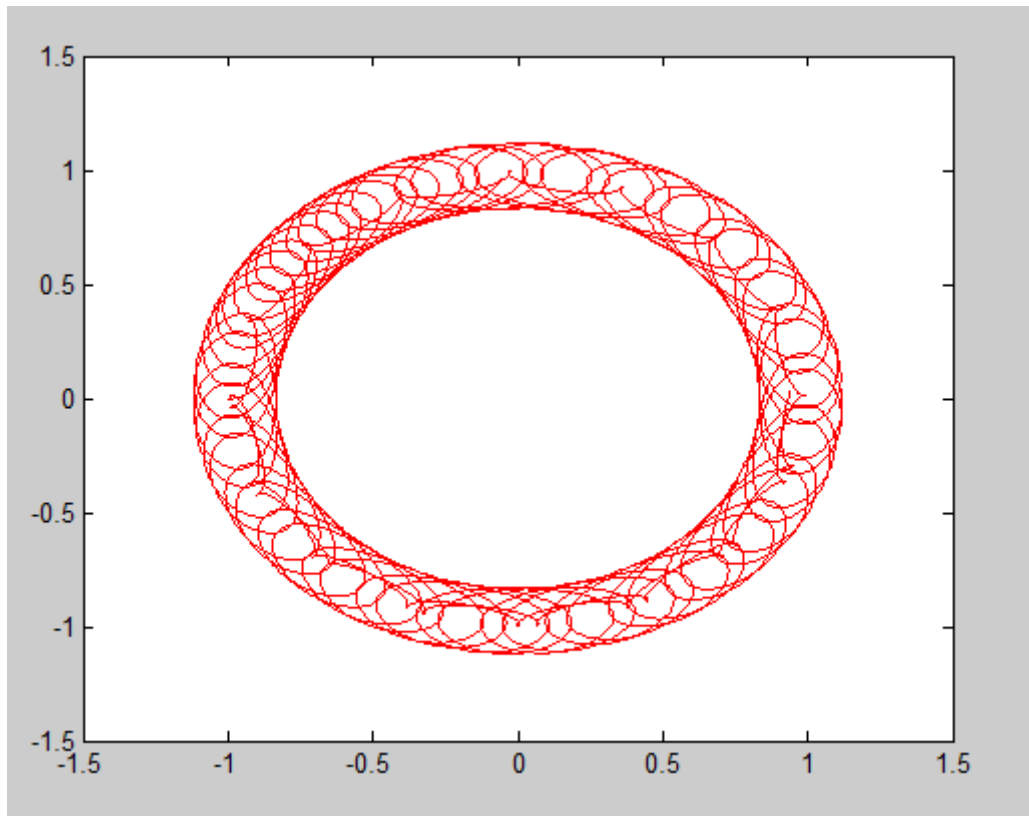


Figura 10: Representació de la trajectòria en el sistema Terra-Sol del punt inicial:  $X = 0.940001252719545$ ,  $Y = 0.03$ ,  $X' = 0.0325229843815117$ ,  $Y' = 0.100095453610786$ .

Fet el breu anàlisi destaquem que ens és indiferent la zona de la figura 8 que estudiem ja que un mateix punt inicial troba talls per les dues zones. Veiem que segons el punt inicial que agafem la trajectòria va variant gradualment, executant una trajectòria més lenta ja que executa voltes més pronunciades.

### 5.1.2 Òrbites amb origen $L_1$ de Mart

En aquest anàlisi per tal d'apreciar millor la òrbita, necessitarem realitzar exploracions per a temps més elevats, com per exemple de cinc-cents unitats. Executem una exploració a partir del programa *ExploracioFinalMart\_2.m* (veure l'apartat A.8.3 de l'annex) per tal de trobar els talls; amb una matriu de punts inicials  $5 \times 5$  (situats en el  $L_1$ ) explorant cinc velocitats per a cada punt inicial i explorant tres constants de Jacobi. Obtenim la figura 11 8 comparant les  $Y$  amb les  $Y'$ :

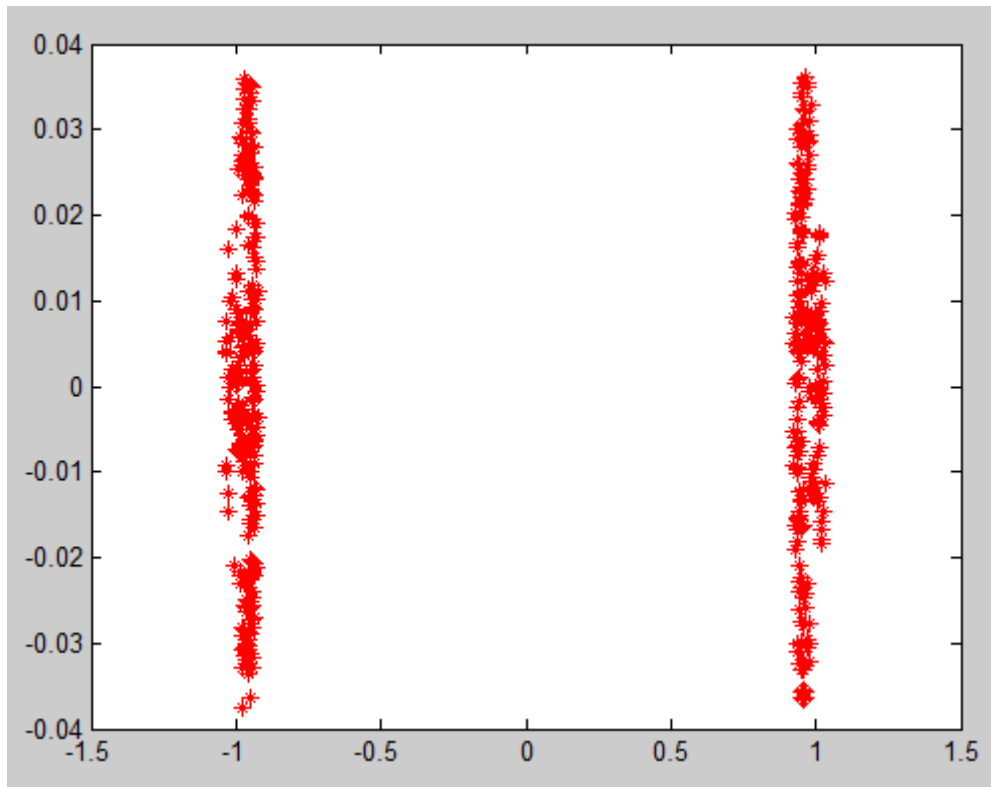


Figura 11: Representació de les posicions de  $Y$  contra la velocitat  $Y'$  per una petita mostra de punts amb origen  $L_1$  de Mart.

Analitzem un parell de punts inicials qualsevols representatiu de l'exploració com per exemple el punt:

$$temps = 0, X = -0.996833915760873, Y = -0.00158288073198358,$$

$$X' = 0.0145439262448666, Y' = 0, C_J = 3$$

I el punt:

$$temps = 0, X = -0.996833915760873, Y = 0, X' = -0.0123511686086668,$$

$$Y' = 0.00897364926476120, C_J = 3$$

A partir dels punts inicials, aplicant el programa *CalculOrbital\_1M.2* (veure l'apartat A.8.1 de l'annex) obtenim la figura 12.

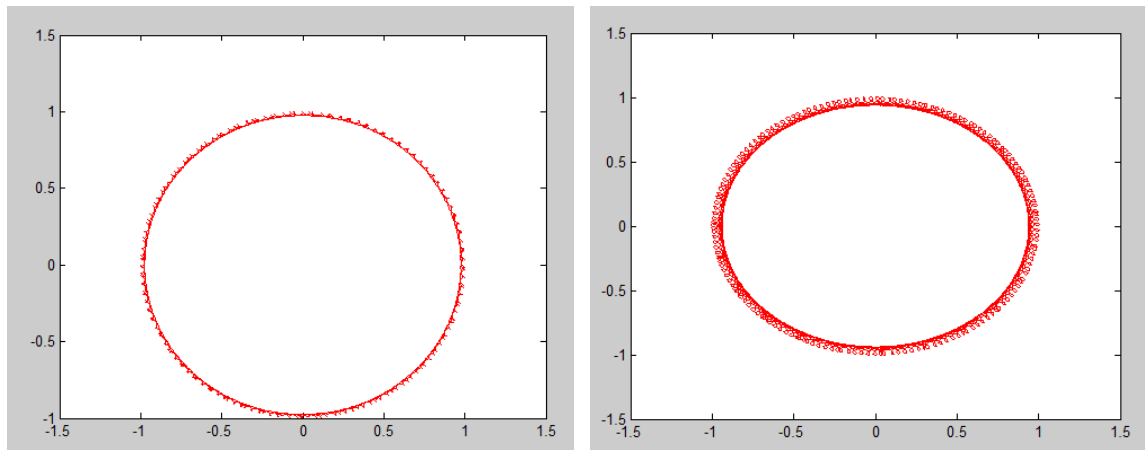


Figura 12: Representació de dues trajectòries en el sistema Mart-Sol.

En aquest cas arribem a una conclusió semblant a la de l'apartat anterior, veiem que ens és indiferent la zona de la figura 11 que estudiem ja que un mateix punt inicial troba talls per a les dues zones. També veiem que segons el punt inicial que agafem la trajectòria va variant gradualment, executant una trajectòria més lenta ja que executa voltes més pronunciades.

## 5.2 Exploració amb origen a $L_1$ de Mart

En primer lloc vam executar el programa *ExploracioFinalMart\_3.m* (veure l'apartat A.8.4 de l'annex) amb una matriu de punts inicials de 50 x 50, amb quatre valors de velocitat per a cada punt, tres constants de Jacobi i per cinc-cents unitats de temps. D'aquesta manera vam explorar un total de trenta mil condicions inicials.

On vam trobar els talls en la secció representats per la figura 13.



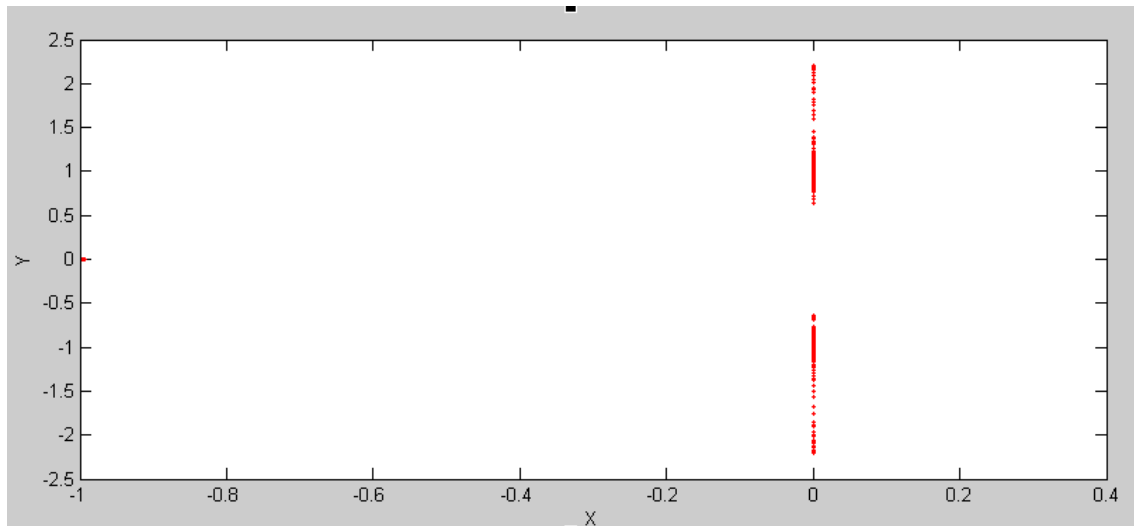


Figura 13: Representació dels punts inicials i els talls en la secció per l'exploració d'origen  $L_1$  de Mart, en el sistema de coordenades Mart-Sol.

Tal com està representat en la figura 13, obtenim talls en la secció per una zona determinada d'aquesta, haurem de comprovar, a partir dels resultats dels següents apartats, si les dues regions on obtenim talls coincideixen.

Tal com hem vist en la figura 13, però aquest cop de forma més detallada, els punts inicials que ens donen aquests talls són els corresponents a la figura 14.

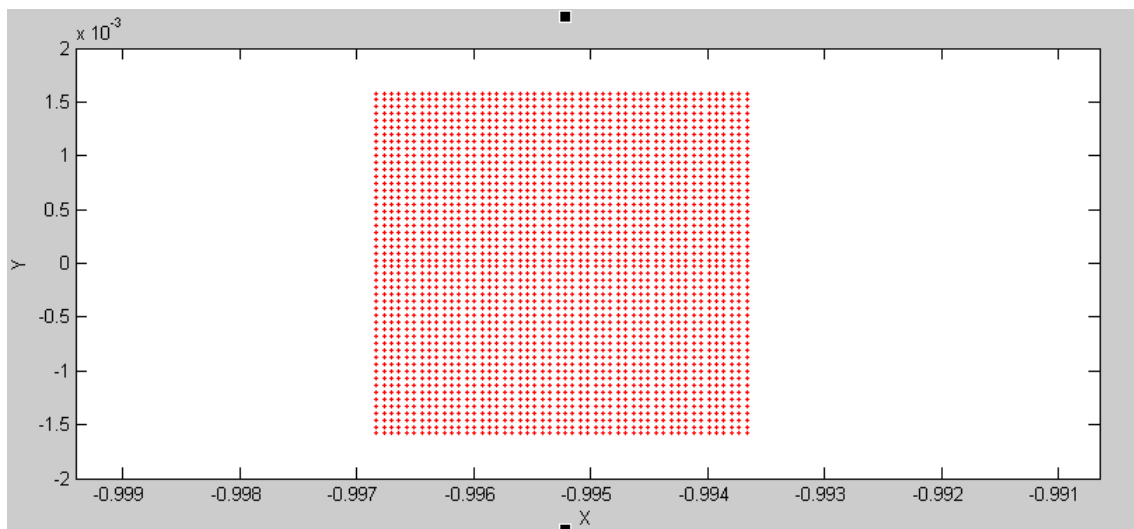


Figura 14: Representació dels punts inicials per l'exploració d'origen  $L_3$  de Mart, en el sistema de coordenades Mart-Sol.

### 5.3 Exploració amb origen a $L_3$ de la Terra

Per tal de realitzar l'exploració vam executar el programa *ExploracioFinalTerra\_3.m* (veure l'apartat A.9.4 de l'annex) amb una matriu de punts inicials de 50 x 50, amb quatre valors de velocitat per a cada punt, però només una constant de Jacobi i per a cinc-cents unitats de temps. Vam explorar un total de deu mil condicions inicials.

On vam trobar els talls en la secció representats en la figura 15.

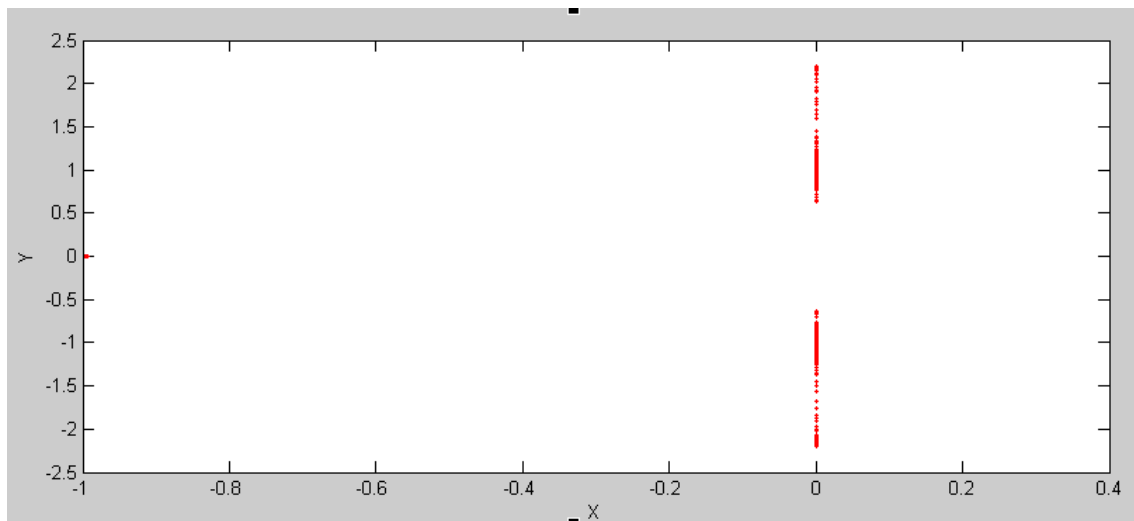


Figura 15: Representació dels punts inicials i els talls en la secció per l'exploració d'origen  $L_3$  de la Terra, en el sistema de coordenades Terra-Sol.

Com veiem en la figura 15, obtenim talls en la secció per una zona determinada, hem de comprovar si aquesta zona coincideix amb la regió determinada en la figura 13. Com hem comentat, cada figura està representada en un sistema diferent, representat per a unitats diferents, de manera que si els volem comparar, haurem de convertir els resultats d'un sistema en unitats de l'altre.

Tal com hem vist en la figura 15, de forma més detallada, els punts inicials que ens donen aquests talls són els corresponents a la figura 16.

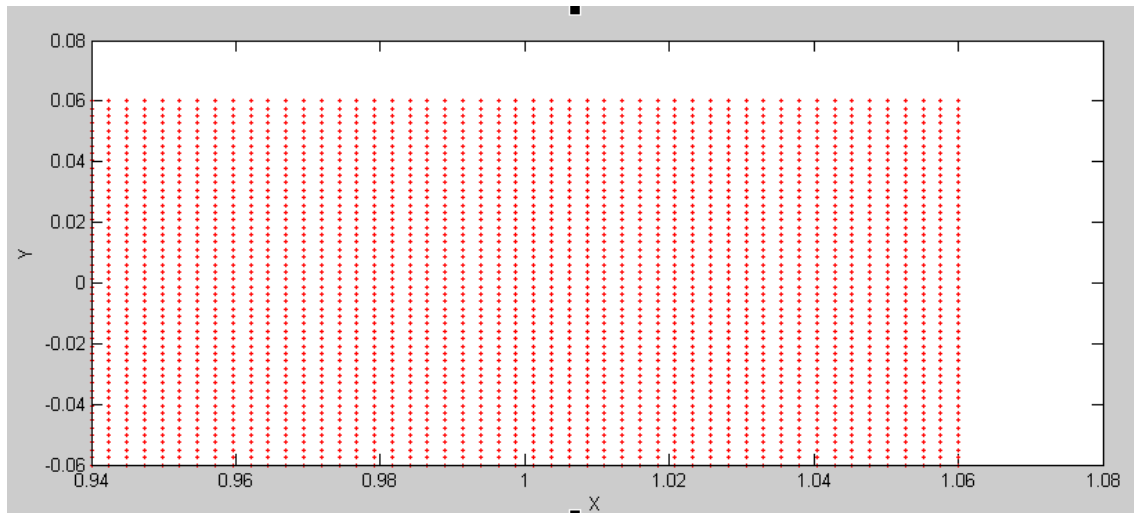


Figura 16: Representació dels punts inicials per l'exploració d'origen  $L_1$  de la Terra, en el sistema de coordenades Terra-Sol.

#### 5.4 Conversió d'unitats del sistema Sol-Mart al sistema Sol-Terra

Per tal de poder contrastar els talls, com hem comentat, hem de convertir els resultats obtinguts en l'exploració del sistema Sol-Mart en unitats del sistema Sol-Terra. Vam aplicar la funció *ConversioM* (veure l'apartat A.10.7 de l'annex), obtenint la figura 17.

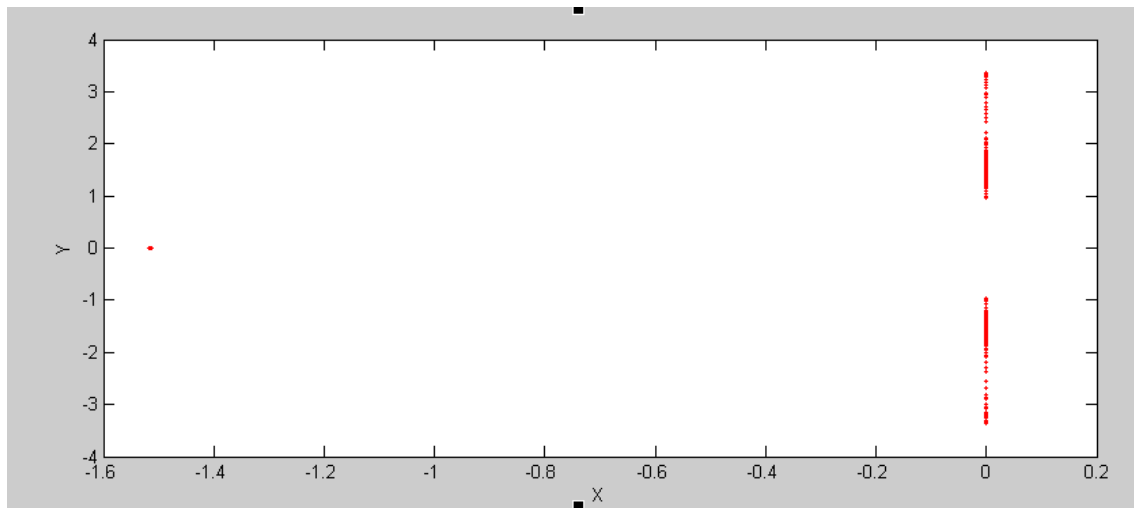


Figura 17: Representació dels punts inicials i els talls en la secció per l'exploració d'origen  $L_3$  de Mart, en el sistema de coordenades Terra-Sol.

D'aquesta manera, arribats a aquest punt podem comparar els resultats expressat en la figura 15 amb els representats en la figura 17, per tal de determinar si existeixen coincidències que ens puguin determinar l'existència d'indícis de transport natural.

### 5.5 Comparació de les dues exploracions

Realitzem una anàlisi per tal de comprovar que tenim coincidències de posicions. Hem barrejat totes les constants de Jacobi ja que cada una representa una velocitat de sortida diferent, tot i que perd el valor conceptual al contrastar els dos sistemes. Contrastem els gràfics obtinguts per un sistema i per a l'altre, i fixant-nos en la zona més interessant obtenim la figura 18.

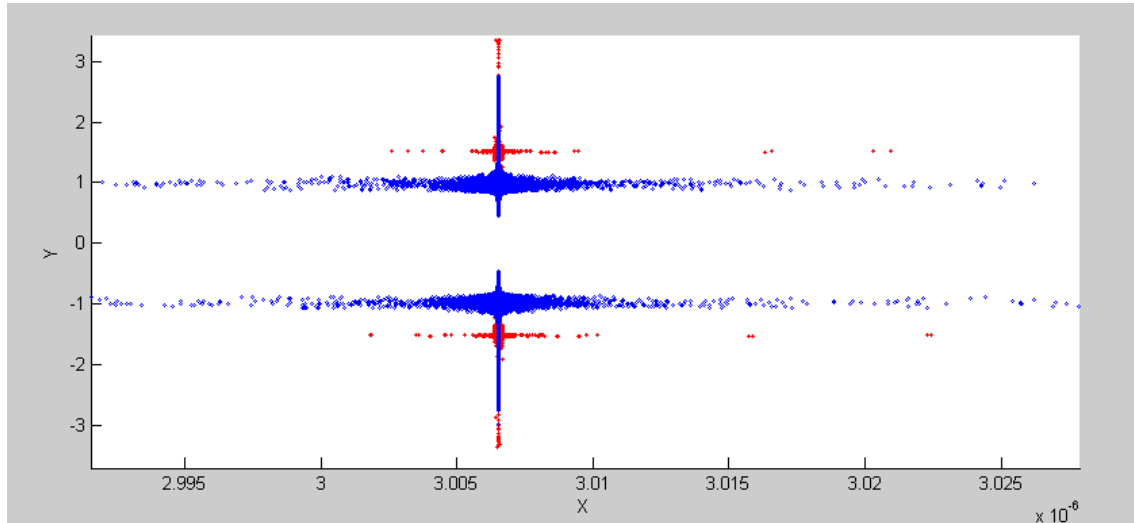
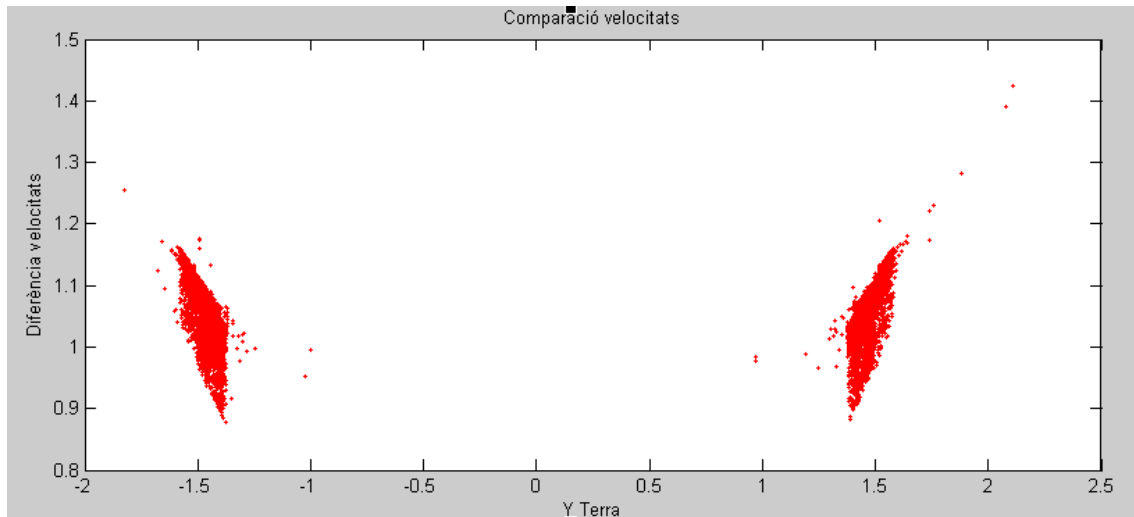


Figura 18: Contrast de la figura 15 (blau) amb la figura 17 (vermell) realitzant un zoom en la zona més interessant.

Com veiem tenim una gran quantitat de possibles punts coincidents, cosa que ens fa pensar que es probable que existeixin indicis de transport natural. Però per a que el transport natural existeixi també hi ha d'haver coincidència en les velocitats.

Per a analitzar si existia coincidència en les velocitats, mitjançant el programa *Comparacio\_3.m* (veure l'apartat A.10.3 de l'annex) hem contrastat els resultats. Com hem comentat en l'apartat 4.5.2 aquest programa ens busca coincidències de posició en la secció amb una precisió de  $10^{-6}$  UA i ens contrasta les diferències de velocitats d'aquests punts coincidents. Així doncs vam obtenir la figura 19.



*Figura 19: Representació de la diferència de velocitats entre els punts coincidents en posició de les dues exploracions.*

Tot i tenir moltes coincidències de posicions, tenim un diferència de velocitats considerables. Hem de tenir en compte que hem executat unes simulacions per un temps de cinc-cents unitats de temps en el sistema Terra-Sol, vindrien a ser uns vuitanta anys, que en termes de transport natural és un temps menyspreable. Vist que estudiant un període tant curt de temps hem obtingut tantes coincidències de posicions és molt probable que realitzant estudis per a temps més prolongats es puguin trobar indicis de transport natural.

## 6 Resum del pressupost

Resum del pressupost general d'execució:

PRESSUPOST D'EXECUCIÓ	8.205	€
14% de despeses generals d'empresa	1.148,7	€
6% de benefici industrial	492,3	€
<b>PRESSUPOST D'EXECUCIÓ TOTAL A FALTA D'IVA</b>	<b>9.846</b>	<b>€</b>

El pressupost general d'execució a falta d'IVA és de nou mil vuit-cents quaranta-sis euros.

## 7 Conclusions

Analitzant els resultats obtinguts en l'apartat 5.5 veiem que tot i tenir moltes coincidències de posicions, tenim una diferència de velocitats considerables. Hem de tenir en compte que hem executat unes simulacions per un temps de cinc-cents unitats de temps en el sistema Terra-Sol, vindrien a ser uns vuitanta anys, que en termes de transport natural és un temps menyspreable.

Per tant podem determinar que a curt termini no tenim indicis de transport natural entre el punt  $L_1$  de Mart i el punt  $L_3$  de la Terra. Però veiem que les diferències de velocitat més petites, tot i ser molt grans per a considerar transport natural es troben de l'ordre de  $10^{-1}$  unitats; que equival a aproximadament a velocitats de tres quilòmetres per segon, velocitats que s'acostumen assolir en maniobres de satèl·lits. És a dir que es pot afirmar que existeixen indicis de transport artificial, ja que podem assolir aquestes diferències de velocitat.

Reafirmant-nos en el fet de que per unes exploracions a curt termini hem sigut capaços de trobar una gran quantitat de coincidències de posicions en la secció (més de vint-i-un mil coincidències), seria interessant realitzar el mateix estudi però a llarg termini. D'aquesta manera, vist el comportament de les trajectòries és molt probable que s'acabin trobant coincidències en posició i velocitat.

També hem de valorar la precisió amb la que hem treballat. Durant l'exercici hem calculat aproximacions amb una precisió de  $10^{-6}$  UA en el cas del sistema Terra-Sol, que són aproximadament cent cinquanta quilòmetres. I cal remarcar que el RTBP no és una representació exacta del sistema Solar, sinó una aproximació, i que d'aquest hem considerat el pla.

Així doncs podem determinar que no hem trobat indicis de transport natural entre els punts  $L_1$  de Mart i el punt  $L_3$  de la Terra a curt termini, tot i que el projecte invita a realitzar un estudi d'indis de existència de transport natural a mitjà o a llarg termini vista la gran quantitat de coincidències de posicions trobades.

Com a propostes de millora, seria interessant realitzar una exploració amb més condicions inicials, ja que degut a que els programes requerien molt de temps a completar-se hem estat limitats.

Es podria estudiar realitzar els programes en una plataforma diferent, de manera que es poguessin optimitzar millor els càlculs a nivell computacional.

També, tal com hem comentat en l'apartat anterior seria d'interès realitzar el mateix estudi però per a un temps més prolongat, ja que en termes de transport natural, els aproximadament vuitanta anys que hem explorat són menyspreables.



## 8 Relació de documents

La relació de documents que componen aquest projecte és:

1- Memòria i Annexes

## 9 Bibliografia

- [1] Ren, Y., Masdemont, J., Gómez, G, Fantino, E..(2012). On the Mechanisms of Natural Transport in the Solar System. *Communications in Nonlinear Science and Numerical Simulations*.
  
- [2] Barrabés, E., Mikkola, S. (2004). Families of periodic horseshoe orbits in the restricted three-body problema. *Astronomy & Astrophysics manuscript*.
  
- [3] Hirsch, M., Smale, S., Devaney, R. (2004). *Differential Equations, Dynamical Systems & An Introduction to Chaos*. Elsevier.
  
- [4] Szebehely, V. (1967). Theory of Orbits. The Restricted Problem of Three Bodies. *Academic Press*.
  
- [5] Danby, J. (1962). *Fundamentals of Celestial Mechanics*. William-Bell.
  
- [6] *Guía de usuario MATLAB*. MathWorks.

## ANNEXOS

### A- Codi informàtic

#### A.1 Funcions

##### A.1.1 FUN\_M

```
function eq = FUN_M(t,x)
%%Equacions desenvolupades:
%Optimització de l'algorisme
%Definició de mu de Mart
mu=3.22775159768e-07;
%Definició dels paràmetres per estalviar càlcul
xm1 = x(1)-mu;
xm2 = x(1)-mu+1;
r12 = xm1^2 +x(2)^2;
r22 = xm2^2 +x(2)^2;
r1 = sqrt(r12);
r2 = sqrt(r22);
r13 = r12*r1;
r23 = r22*r2;
mul = mu - 1;
%Equacions:
eq=[x(3);x(4); 2*x(4)-(mu*xm2)/r23 + mul*xm1/r13 + x(1); -2*x(3) -
mu*x(2)/r22 + mul*x(2)/r13 + x(2)];
end
```

##### A.1.2 FUN\_T

```
function eq = FUN_T(t,x)
%%Equacions desenvolupades:
%Optimització de l'algorisme
%Definició de mu de la Terra
mu=3.00652690848e-06;
%Definició dels paràmetres per estalviar càlcul
xm1 = x(1)-mu;
xm2 = x(1)-mu+1;
r12 = xm1^2 +x(2)^2;
r22 = xm2^2 +x(2)^2;
r1 = sqrt(r12);
r2 = sqrt(r22);
r13 = r12*r1;
r23 = r22*r2;
mul = mu - 1;
%Equacions:
eq=[x(3);x(4); 2*x(4)-(mu*xm2)/r23 + mul*xm1/r13 + x(1); -2*x(3) -
mu*x(2)/r22 + mul*x(2)/r13 + x(2)];
end
```

#### A.2 Constant de Jacobi

##### A.2.1 CJ\_Y

```
function [Cj_Y,Ybona]=CJ_Y(x,mu,Cj)
% Càlcul de la velocitat de Y a partir de la Constant de Jacobi i les
% posicions de X, Y i la velocitat de X.
% Definició dels paràmetres per estalviar càlcul
xm1 = x(1)-mu;
xm2 = x(1)-mu+1;
r12 = xm1^2 +x(2)^2;
r22 = xm2^2 +x(2)^2;
```

```

r1 = sqrt(r12);
r2 = sqrt(r22);
%Càlcul final
yp=-Cj+x(1)^2+x(2)^2+2*(1-mu)/r1 +2*mu/r2+mu*(1-mu)-x(3)^2;
if yp < 0
    Ybona=0;
end
Cj_Y=sqrt(yp);
Ybona=1;
end

```

#### A.2.2 CJ

```

function Cj=CJ(x,mu)
%Càlcul de la constant de Jacobi a partir de la posició i la
velocitat.
%Definició dels paràmetres per estalviar càlcul
xm1 = x(1)-mu;
xm2 = x(1)-mu+1;
r12 = xm1^2 +x(2)^2;
r22 = xm2^2 +x(2)^2;
r1 = sqrt(r12);
r2 = sqrt(r22);
%Càlcul final
Cj=x(1)^2+x(2)^2+2*(1-mu)/r1 +2*mu/r2+mu*(1-mu)-x(3)^2-x(4)^2;
end

```

### A.3 Generació de punts inicials

#### A.3.1 PI\_M

```

function PI=PI_M(Num,NumAngles,Cj)
%Funció que a partir del Num (numero de files/columnes de la matriu de
%punts inicials) i el NumAngles que volem explorar del vector de
velocitats
%ens dóna una matriu de punts inicials [x(1),x(2),x(3),x(4)]
%En aquest cas generem els punts inicials de MART:
mu=3.22775159768e-07;
Li=L1(mu);
D=(Li-(mu-1))/3;
M=GeneracioM_2(Li,D,Num);
[filesM,columnesM]=size(M);
contador=1;
while contador<=filesM
    x=M(contador,:);
    X_1=GeneracioPI(Cj,mu,NumAngles,x);
    if contador==1
        X=X_1;
    else
        X=[X;X_1];
    end
    contador=contador+1;
end
PI=X;
end

```

#### A.3.2 PI\_T

```

function PI=PI_T(D,Num,NumAngles,Cj)
%Funció que a partir del Num (numero de files/columnes de la matriu de
%punts inicials) i el NumAngles que volem explorar del vector de
velocitats
%ens dóna una matriu de punts inicials [x(1),x(2),x(3),x(4)]
%On D és la distància del rang de Punts Inicials respecte L3

```

```
%En aquest cas generem els punts inicials de la TERRA:
```

```
mu=3.00652690848e-06;
Li=L3(mu);
M=GeneracioM_2(Li,D,Num);
[filesM,columnesM]=size(M);
contador=1;
while contador<=filesM
    x=M(contador,:);
    X_1=GeneracioPI(Cj,mu,NumAngles,x);
    if contador==1
        X=X_1;
    else
        X=[X;X_1];
    end
    contador=contador+1;
end
PI=X;
end
```

### A.3.3 GeneracioM\_2

```
function Matriu=GeneracioM_2(Li,D,Num)
%Funció que a partir del punt Li demanat i la distància a ell, genera
un
%vector on la primera columna serà x(1) i la segona x(2)
Increment=2*D/(Num-1);
x(1)=Li-D;
x(2)=-D;
i=1;
j=1;
k=1;
while k<=2*Num
    while i<=Num
        while j<=Num
            Matriu(k,1)=x(1);
            Matriu(k,2)=x(2);
            x(2)=x(2)+Increment;
            k=k+1;
            j=j+1;
        end
        j=1;
        x(2)=-D;
        x(1)=x(1)+Increment;
        i=i+1;
    end
end
end
```

### A.3.4 L1

```
function l1=L1(mu)
l1=-1+(mu/3)^(1/3)-(1/3)*(mu/3)^(2/3)+(26/9)*(mu/3);
end
```

### A.3.5 L3

```
function l3=L3(mu)
l3=1+(5/12)*mu;
end
```

### A.3.6 Determinació de la distància

```
%ALGORISME PER A DETERMINAR UNA D CORRECTE PER A LA TERRA
mu=3.00652690848e-06;
%Definició del temps:
```

```

tspan=[0,-1000];
%Valors inicials
x1=L3(mu);
x1=x1-0.06;
x2=0;
x3=0;
Cj=3;
x=[x1,x2,x3];
x4=CJ_Y(x,mu,Cj);
x0=[x1,x2,x3,x4];
%Tipus de ode entrada:
ode_fcn_format=0;
%Entrem el valor de la tolerància:
tol = 1.e-6;
%Volem que es guardin tots els passos? si:trace=1, no:trace=0
trace=0;
%Contador no se per a què serveix:
count=0;
%Pas(step) màxim:
hmax=0.1;
%%Iniciem l'algorisme
options=odeset('MaxStep',hmax,'RelTol',tol);
[T,X]=ode45('FUN_T',tspan,x0,options);
%Mostrar els resultats
plot(X(:,1),X(:,2),'r')
[T,X];

```

#### A.3.7 GeneracioPI

```

function PuntsIni=GeneracioPI(Cj,mu,NumAngles,x)
%Funció que a partir de una Constant de Jacobi i un punt
x=[x(1),x(2)] en
%calcula la velocitat en mòdul i li dóna un valor angular NumAngles
vegades
%per a generar una matriu de PuntsInicials.
xm1 = x(1)-mu;
xm2 = x(1)-mu+1;
r12 = xm1^2 +x(2)^2;
r22 = xm2^2 +x(2)^2;
r1 = sqrt(r12);
r2 = sqrt(r22);
V=(x(1)^2+x(2)^2+2*(1-mu)/r1 +2*mu/r2+mu*(1-mu)-Cj);
if V < 0
    text=['La Cj: ',num2str(Cj),' es incorrecte per el punt estudiat:
',num2str(x)];
    disp(text)
    PuntsIni=[0,0,0,0];
    return
end
V=V^(1/2);
Increment=2*pi/NumAngles;
angle=0;
contador=0;
while angle<2*pi
    x(3)=V*cos(angle);
    x(4)=V*sin(angle);
    angle=angle+Increment;
    contador=contador+1;
    B(contador,:)=x;
end
PuntsIni=B;
end

```

## A.3.8 ProvaD

```
%ALGORISME PER A DETERMINAR UNA D CORRECTE PER A LA TERRA
mu=3.00652690848e-06;
%Definició del temps:
tspan=[0,-1000];
%Valors inicials
x1=L3(mu);
x1=x1-0.06;
x2=0;
x3=0;
Cj=3;
x=[x1,x2,x3];
x4=CJ_Y(x,mu,Cj);
x0=[x1,x2,x3,x4];
%Tipus de ode entrada:
ode_fcn_format=0;
%Entrem el valor de la tolerància:
tol = 1.e-6;
%Volem que es guardin tots els passos? si:trace=1, no:trace=0
trace=0;
%Contador no se per a què serveix:
count=0;
%Pas(step) màxim:
hmax=0.1;
%%Iniciem l'algorisme
options=odeset('MaxStep',hmax,'RelTol',tol);
[T,X]=ode45('FUN_T',tspan,x0,options);
%Mostrar els resultats
plot(X(:,1),X(:,2),'r')
[T,X];
```

## A.4 ode78

```
function [tout,xout] =
ode78(FUN,tspan,x0,ode_fcn_format,tol,trace,count,hmax)

% Copyright (C) 2001, 2000 Marc Compere
% This file is intended for use with Octave.
% ode78.m is free software; you can redistribute it and/or modify it
% under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2, or (at your option)
% any later version.
%
% ode78.m is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details at
% www.gnu.org/copyleft/gpl.html.
%
% -----
%
% ode78 (v1.14) Integrates a system of ordinary differential equations
% using
% 7th order formulas.
%
% This is a 7th-order accurate integrator therefore the local error
% normally
% expected is  $O(h^8)$ . However, because this particular implementation
% uses the 8th-order estimate for xout (i.e. local extrapolation)
% moving
```

```

% forward with the 8th-order estimate will yield errors on the order
of  $O(h^9)$ .
%
% The order of the RK method is the order of the local *truncation*
error,  $d$ ,
% which is the principle error term in the portion of the Taylor
series
% expansion that gets dropped, or intentionally truncated. This is
different
% from the local error which is the difference between the estimated
solution
% and the actual, or true solution. The local error is used in
stepsize
% selection and may be approximated by the difference between two
estimates of
% different order,  $l(h) = x_{(O(h+1))} - x_{(O(h))}$ . With this
definition, the
% local error will be as large as the error in the lower order method.
% The local truncation error is within the group of terms that gets
multiplied
% by  $h$  when solving for a solution from the general RK method.
Therefore, the
% order- $p$  solution created by the RK method will be roughly accurate
to  $O(h^{(p+1)})$ 
% since the local truncation error shows up in the solution as  $h*d$ ,
which is
%  $h$  times an  $O(h^p)$  term, or rather  $O(h^{(p+1)})$ .
% Summary: For an order- $p$  accurate RK method,
%           - the local truncation error is  $O(h^p)$ 
%           - the local error used for stepsize adjustment and that
%             is actually realized in a solution is  $O(h^{(p+1)})$ 
%
% This requires 13 function evaluations per integration step.
%
% Relevant discussion on step size choice can be found on pp.90,91 in
% U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary
Differential Equations
% and Differential-Algebraic Equations, Society for Industrial and
Applied Mathematics
% (SIAM), Philadelphia, 1998
%
% More may be found in the original author's text containing numerous
% applications on ordinary and partial differential equations using
Matlab:
%
% Howard Wilson and Louis Turcotte, 'Advanced Mathematics and
Mechanics Applications Using MATLAB', 2nd Ed, CRC Press, 1997
%
%
%           [tout, xout] =
ode78(FUN,tspan,x0,ode_fcn_format,tol,trace,count,hmax)
%
% INPUT:
% FUN - String containing name of user-supplied problem description.
%       Call:  $x_{prime} = fun(t,x)$  where  $FUN = 'fun'$ .
%        $t$  - Time (scalar).
%        $x$  - Solution column-vector.
%        $x_{prime}$  - Returned derivative COLUMN-vector;  $x_{prime}(i) =$ 
 $dx(i)/dt$ .
% tspan - [ tstart, tfinal ]
% x0 - Initial value COLUMN-vector.

```



```

% ode_fcn_format - this specifies if the user-defined ode function is
in
%               the form:      xprime = fun(t,x)      (ode_fcn_format=0,
default)
%               or:            xprime = fun(x,t)      (ode_fcn_format=1)
%               Matlab's solvers comply with ode_fcn_format=0 while
%               Octave's lsode() and sdirk4() solvers comply with
ode_fcn_format=1.
% tol - The desired accuracy. (optional, default: tol = 1.e-6).
% trace - If nonzero, each step is printed. (optional, default: trace
= 0).
% count - if nonzero, variable 'rhs_counter' is initialized, made
global
%           and counts the number of state-dot function evaluations
%           'rhs_counter' is incremented in here, not in the state-dot
file
%           simply make 'rhs_counter' global in the file that calls
ode78
% hmax - limit the maximum stepsize to be less than or equal to hmax
%
% OUTPUT:
% tout - Returned integration time points (row-vector).
% xout - Returned solution, one solution column-vector per tout-
value.
%
% The result can be displayed by: plot(tout, xout).

% Daljeet Singh & Howard Wilson
% Dept. Of Electrical Engg., The University of Alabama.
% 11-24-1988.
%
% modified by:
% Marc Compere
% CompereM@asme.org
% created : 06 October 1999
% modified: 19 May 2001

% The Fehlberg coefficients:
alpha_ = [ 2./27., 1/9, 1/6, 5/12, 0.5, 5/6, 1/6, 2/3, 1/3, 1, 0, 1
]';
beta_ = [ 2/27, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;
1/36, 1/12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;
1/24, 0, 1/8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;
5/12, 0, -25/16, 25/16, 0, 0, 0, 0, 0, 0, 0, 0, 0 ;
0.05, 0, 0, 0.25, 0.2, 0, 0, 0, 0, 0, 0, 0, 0 ;
-25/108, 0, 0, 125/108, -65/27, 125/54, 0, 0, 0, 0, 0, 0, 0
;
31/300, 0, 0, 0, 61/225, -2/9, 13/900, 0, 0, 0, 0, 0, 0 ;
2, 0, 0, -53/6, 704/45, -107/9, 67/90, 3, 0, 0, 0, 0, 0 ;
-91/108, 0, 0, 23/108, -976/135, 311/54, -19/60, 17/6, -
1/12, 0, 0, 0, 0, 0 ;
2383/4100, 0, 0, -341/164, 4496/1025, -301/82, 2133/4100,
45/82, 45/164, 18/41, 0, 0, 0 ;
3/205, 0, 0, 0, 0, -6/41, -3/205, -3/41, 3/41, 6/41, 0, 0, 0
;
-1777/4100, 0, 0, -341/164, 4496/1025, -289/82, 2193/4100,
51/82, 33/164, 12/41, 0, 1, 0 ]';
chi_ = [ 0, 0, 0, 0, 0, 34/105, 9/35, 9/35, 9/280, 9/280, 0, 41/840,
41/840]';
psi_ = [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, -1 ]';

```

```

pow = 1/8; % see p.91 in the Ascher & Petzold reference for more
information.

if nargin < 8, hmax = (tspan(2) - tspan(1))/2.5; end
if nargin < 7, count = 0; end
if nargin < 6, trace = 0; end
if nargin < 5, tol = 1.e-6; end
if nargin < 4, ode_fcn_format = 0; end

% Initialization
t0 = tspan(1);
tfinal = tspan(2);
t = t0;
% the following step parameters are used in ODE45
% hmax = (tfinal - t)/5;
% hmin = (tfinal - t)/20000;
% h = (tfinal - t)/100;
% The following parameters were taken because the integrator has
% higher order than ODE45. This choice is somewhat subjective.
%hmin = (tfinal - t)/10000;
hmin = (tfinal - t)/1e20;
h = (tfinal - t)/50;
x = x0(:); % the ':' ensures x is initialized as a column
vector
f = x*zeros(1,13); % f needs to be an Nx13 matrix where N=number of
rows in x
tout = t;
xout = x.';
tau = tol * max(norm(x,'inf'), 1);

if count==1,
    global rhs_counter
    if ~exist('rhs_counter'),rhs_counter=0; end
end % if count

if trace
    % clc, t, h, x
    % clc, t, x
    clc, t
end

% The main loop
while (t < tfinal) && (h >= hmin)
    if t + h > tfinal, h = tfinal - t; end

    % Compute the slopes
    if (ode_fcn_format==0), % (default)
        f(:,1) = feval(FUN,t,x);
        for j = 1: 12,
            f(:,j+1) = feval(FUN, t+alpha_(j)*h, x+h*f*beta_(:,j));
        end
    else % ode_fcn_format==1
        f(:,1) = feval(FUN,x,t);
        for j = 1: 12,
            f(:,j+1) = feval(FUN, x+h*f*beta_(:,j), t+alpha_(j)*h);
        end
    end % if (ode_fcn_format==1)

```

```

% increment rhs_counter
if count==1, rhs_counter = rhs_counter + 13; end

% Truncation error term
gamma1 = h*41/840*f*psi_;

% Estimate the error and the acceptable error
delta = norm(gamma1,'inf');
tau = tol*max(norm(x,'inf'),1.0);

% Update the solution only if the error is acceptable
if delta <= tau
    t = t + h;
    x = x + h*f*chi_; % this integrator uses local extrapolation
    tout = [tout; t];
    xout = [xout; x.'];
end
if trace
    home, t, h, x
    home, t, x
    home, t, h
end

% Update the step size
if delta == 0.0
    delta = 1e-16;
end
h = min(hmax, 0.8*h*(tau/delta)^pow);

end;

if (t < tfinal)
    disp('SINGULARITY LIKELY.')
    t
end

```

## A.5 Newton-Rapshon

### A.5.1 Newton

```

function N=Newton(A1,A2)
%Desenvolupament de l'algorisme de Newton-Rapson per a trobar talls en
y=0.
%On N serà el punt final, de y=0.
%A1 i A2 són els punts inicial i final
%tol_1 és la tolerància en que acceptem el zero
%Per a poder executar-lo es necessiten el document "FUN.m".
NewtonMax=0;
tol_1=1.e-8;
%Assignem el punt inicial per el càlcul
if abs(A1(5))>abs(A2(5)) %Comparem la derivada per a que els càlculs
siguin més ràpids
An=A1;
else An=A2;
end
%Desenvolupem la iteració
while abs(An(3))>tol_1 && NewtonMax < 15 %%Fem màxim 15 iteracions
    hmax_1=-An(3)/An(5);
    tspan_1=[An(1),An(1)+hmax_1];
    x0_1=[An(2),An(3),An(4),An(5)];

```

```

options_1=odeset('MaxStep',hmax_1,'RelTol',tol_1);
[tout,xout]=ode45('FUN',tspan_1,x0_1,options_1);
An=[tout,xout];
[filesAn,columnesAn]=size(An);
An=An(filesAn,:);
NewtonMax=NewtonMax+1;
end
N=An;
end

```

#### A.5.2 *Newton\_M*

```

function N=Newton_M(A1,A2)
%Desenvolupament de l'algorisme de Newton-Rapson per a trobar talls en
y=0.
%On N serà el punt final, de y=0.
%A1 i A2 són els punts inicial i final
%tol_1 és la tolerància en que acceptem el zero
%Per a poder executar-lo es necessiten el document "FUN.m".
NewtonMax=0;
tol_1=1.e-8;
%Assignem el punt inicial per el càlcul
if abs(A1(5))>abs(A2(5)) %Comparem la derivada per a que els càlculs
siguin més ràpids
An=A1;
else An=A2;
end
%Desenvolupem la iteració
while abs(An(2))>tol_1 && NewtonMax < 15 %%Fem màxim 15 iteracions
    hmax_1=-An(2)/An(4);
    if hmax_1==0
        break
    end
    tspan_1=[An(1),An(1)+hmax_1];
    x0_1=[An(2),An(3),An(4),An(5)];
    options_1=odeset('MaxStep',hmax_1,'RelTol',tol_1);
    [tout,xout]=ode45('FUN_M',tspan_1,x0_1,options_1);
    An=[tout,xout];
    [filesAn,columnesAn]=size(An);
    An=An(filesAn,:);
    NewtonMax=NewtonMax+1;
end
N=An;
end

```

#### A.5.3 *Newton\_T*

```

function N=Newton_T(A1,A2)
%Desenvolupament de l'algorisme de Newton-Rapson per a trobar talls en
y=0.
%On N serà el punt final, de y=0.
%A1 i A2 són els punts inicial i final
%tol_1 és la tolerància en que acceptem el zero
%Per a poder executar-lo es necessiten el document "FUN.m".
NewtonMax=0;
tol_1=1.e-8;
%Assignem el punt inicial per el càlcul
if abs(A1(5))>abs(A2(5)) %Comparem la derivada per a que els càlculs
siguin més ràpids
An=A1;
else An=A2;
end
%Desenvolupem la iteració

```

```

while abs(An(2))>tol_1 && NewtonMax < 15 %%Fem màxim 15 iteracions
    hmax_1=-An(2)/An(4);
    if hmax_1==0
        break
    end
    tspan_1=[An(1),An(1)+hmax_1];
    x0_1=[An(2),An(3),An(4),An(5)];
    options_1=odeset('MaxStep',hmax_1,'RelTol',tol_1);
    [tout,xout]=ode45('FUN_T',tspan_1,x0_1,options_1);
    An=[tout,xout];
    [filesAn,columnesAn]=size(An);
    An=An(filesAn,:);
    NewtonMax=NewtonMax+1;
end
N=An;
end

```

#### A.5.4 Newton\_MS

```

function N=Newton_MS(A1,A2)
%Desenvolupament de l'algorisme de Newton-Rapson per a trobar talls en
y=0.
%On N serà el punt final, de y=0.
%A1 i A2 són els punts inicial i final
%tol_1 és la tolerància en que acceptem el zero
%Per a poder executar-lo es necessiten el document "FUN.m".
NewtonMax=0;
tol_1=1.e-10;
mu=3.22775159768e-07;
%Assignem el punt inicial per el càlcul
if abs(A1(5))>abs(A2(5)) %Comparem la derivada per a que els càlculs
siguin més ràpids
    An=A1;
else An=A2;
end
%Desenvolupem la iteració
while (abs(An(2))>tol_1+mu || abs(An(2))<=-tol_1+mu)&& NewtonMax < 15
%%Fem màxim 15 iteracions
    hmax_1=-An(2)/An(4);
    if hmax_1==0
        break
    end
    tspan_1=[An(1),An(1)+hmax_1];
    x0_1=[An(2),An(3),An(4),An(5)];
    options_1=odeset('MaxStep',hmax_1,'RelTol',tol_1);
    [tout,xout]=ode45('FUN_M',tspan_1,x0_1,options_1);
    An=[tout,xout];
    [filesAn,columnesAn]=size(An);
    An=An(filesAn,:);
    An(2)=An(2)+mu;
    NewtonMax=NewtonMax+1;
end
N=An;
end

```

#### A.5.5 Newton\_TS

```

function N=Newton_TS(A1,A2)
%Desenvolupament de l'algorisme de Newton-Rapson per a trobar talls en
y=0.
%On N serà el punt final, de y=0.
%A1 i A2 són els punts inicial i final
%tol_1 és la tolerància en que acceptem el zero

```

```

%Per a poder executar-lo es necessiten el document "FUN.m".
NewtonMax=0;
tol_1=1.e-10;
mu=3.00652690848e-06;
%Assignem el punt inicial per el càlcul
if abs(A1(5))>abs(A2(5)) %Comparem la derivada per a que els càlculs
siguin més ràpids
An=A1;
else An=A2;
end
%Desenvolupem la iteració
while abs(An(2))>tol_1 && NewtonMax < 15 %%Fem màxim 15 iteracions
    hmax_1=-An(2)/An(4);
    if hmax_1==0
        break
    end
    tspan_1=[An(1),An(1)+hmax_1];
    x0_1=[An(2),An(3),An(4),An(5)];
    options_1=odeset('MaxStep',hmax_1,'RelTol',tol_1);
    [tout,xout]=ode45('FUN_T',tspan_1,x0_1,options_1);
    An=[tout,xout];
    [filesAn,columnesAn]=size(An);
    An=An(filesAn,:);
    An(2)=An(2)+mu;
    NewtonMax=NewtonMax+1;
end
N=An;
end

```

## A.6 Càlcul de la secció

### A.6.1 Seccio

```

function Sec=Seccio(A1,A2)
%Funció que a partir de dos punts determina si la seva unió a través
de les
%equacions "FUN" talla una determinada secció
%En aquest cas la secció és X=0
Sec=A1(2)*A2(2);
end

```

### A.6.2 SeccioMS

```

function Sec=SeccioMS(A1,A2)
%Funció que a partir de dos punts determina si la seva unió a través
de les
%equacions "FUN" talla una determinada secció
%En aquest cas la secció és X=0
mu=3.22775159768e-07;
Sec=(A1(2)+mu)*(A2(2)+mu);
end

```

### A.6.3 SeccioTS

```

function Sec=SeccioTS(A1,A2)
%Funció que a partir de dos punts determina si la seva unió a través
de les
%equacions "FUN" talla una determinada secció
%En aquest cas la secció és X=0
mu=3.00652690848e-06;
Sec=(A1(2)+mu)*(A2(2)+mu);
end

```

## A.7 Càlcul dels Talls

## A.7.1 TallsT

```

function B=TallsT(TempsFi,ht,x0,Nt,tol)
%Funció que a partir d'un rang de temps que comença a 0 fins a TempsFi
%(explorant en salts de ht, és a dir l'increment de temps entre un
punt i el següent és ht)
%explora la òrbita que comença en el punt x0 fins que en troba Nt
talls a
%l'eix y=0 amb una tolerància tol_1.
%Per a poder executar-lo es necessiten els documents "FUN.m",
"Newton.m".
%Definició del temps
Ti=0;
Tf=ht;
tspan=[Ti,Tf];
Ntotal=0;
B_1=0;
B_2=x0;
B(1,:)= [B_1,B_2];
options=odeset('MaxStep',ht,'RelTol',tol);
A1=[0,x0];
while Ntotal<Nt
    while Tf<TempsFi
        %Algorisme de càlcul
        tspan=[Ti,Tf];
        [T,X]= ode45('FUN',tspan,x0,options);
        %Donem un nom més fàcil a la matriu per a poder treballar-la
        A_1=[T,X];
        [filesA_1,columnesA_1]=size(A_1);
        %D'aquesta manera A serà un vector que contindrà el següent
punt de la iteració
        A2=A_1(filesA_1,:);
        if A1(3)*A2(3) < 0 %Si les Y tenen signe diferent apliquem
Newton
            N=Newton(A1,A2);
            Ntotal=Ntotal+1;
            B(Ntotal+1,:)=N;
        end
        if Ntotal==Nt
            break
        end
        A1=A2;
        x0=[A1(2),A1(3),A1(4),A1(5)];
        Ti=Tf;
        Tf=Tf+ht;
    end
    if Tf<TempsFi
    else break
    end
end
end

```

## A.7.2 TallsT\_M

```

function B=TallsT_M(TempsFi,ht,x0,tol)
%Funció que a partir d'un rang de temps que comença a 0 fins a TempsFi
%(explorant en salts de ht, és a dir l'increment de temps entre un
punt i el següent és ht)
%explora la òrbita que comença en el punt x0 fins que en troba els
talls a

```

```

%l'eix y=0 amb una tolerància tol_1.
%Per a poder executar-lo es necessiten els documents "FUN.m",
"Newton.m".
%Definició del temps
Ti=0;
Tf=ht;
tspan=[Ti,Tf];
Ntotal=0;
B_1=0;
B_2=x0;
B(1,:)=[B_1,B_2];
options=odeset('MaxStep',ht,'RelTol',tol);
A1=[0,x0];
while Tf<=TempsFi
    %Algorisme de càlcul
    tspan=[Ti,Tf];
    [T,X]= ode45('FUN_M',tspan,x0,options);
    %Donem un nom més fàcil a la matriu per a poder treballar-la
    A_1=[T,X];
    [filesA_1,columnesA_1]=size(A_1);
    %D'aquesta manera A serà un vector que contindrà el següent punt
    de la iteració
    A2=A_1(filesA_1,:);
    Sec=Seccio(A1,A2);
    if Sec < 0 %Si les Y tenen signe diferent apliquem Newton
        N=Newton_M(A1,A2);
        Ntotal=Ntotal+1;
        B(Ntotal+1,:)=N;
    end
    A1=A2;
    x0=[A1(2),A1(3),A1(4),A1(5)];
    Ti=Tf;
    Tf=Tf+ht;
end
end

```

### A.7.3 TallsT\_T

```

function B=TallsT_T(TempsFi,ht,x0,tol)
%Funció que a partir d'un rang de temps que comença a 0 fins a TempsFi
%(explorant en salts de ht, és a dir l'increment de temps entre un
punt i el següent és ht)
%explora la òrbita que comença en el punt x0 fins que en troba els
talls a
%l'eix y=0 amb una tolerància tol_1.
%Per a poder executar-lo es necessiten els documents "FUN.m",
"Newton.m".
%Definició del temps
Ti=0;
Tf=-ht;
tspan=[Ti,Tf];
Ntotal=0;
B_1=0;
B_2=x0;
B(1,:)=[B_1,B_2];
options=odeset('MaxStep',ht,'RelTol',tol);
A1=[0,x0];
while Tf>=TempsFi
    %Algorisme de càlcul
    tspan=[Ti,Tf];
    [T,X]= ode45('FUN_T',tspan,x0,options);
    %Donem un nom més fàcil a la matriu per a poder treballar-la

```



```

    A_1=[T,X];
    [filesA_1,columnesA_1]=size(A_1);
    %D'aquesta manera A serà un vector que contindrà el següent
punt de la iteració
    A2=A_1(filesA_1,:);
    Sec=Seccio(A1,A2);
    if Sec < 0 %Si les Y tenen signe diferent apliquem Newton
        N=Newton_T(A1,A2);
        Ntotal=Ntotal+1;
        B(Ntotal+1,:)=N;
    end
    A1=A2;
    x0=[A1(2),A1(3),A1(4),A1(5)];
    Ti=Tf;
    Tf=Tf-ht;
end
end

```

#### A.7.4 TallsT\_MS

```

function B=TallsT_MS(TempsFi,ht,x0,tol)
%Funció que a partir d'un rang de temps que comença a 0 fins a TempsFi
%(explorant en salts de ht, és a dir l'increment de temps entre un
punt i el següent és ht)
%explora la òrbita que comença en el punt x0 fins que en troba els
talls a
%l'eix y=0 amb una tolerància tol_1.
%Per a poder executar-lo es necessiten els documents "FUN.m",
"Newton.m".
%Definició del temps
Ti=0;
Tf=ht;
tspan=[Ti,Tf];
Ntotal=0;
B_1=0;
B_2=x0;
B(1,:)=[B_1,B_2];
options=odeset('MaxStep',ht,'RelTol',tol);
A1=[0,x0];
while Tf<=TempsFi
    %Algorisme de càlcul
    tspan=[Ti,Tf];
    [T,X]=ode45('FUN_M',tspan,x0,options);
    %Donem un nom més fàcil a la matriu per a poder treballar-la
    A_1=[T,X];
    [filesA_1,columnesA_1]=size(A_1);
    %D'aquesta manera A serà un vector que contindrà el següent punt
de la iteració
    A2=A_1(filesA_1,:);
    Sec=SeccioMS(A1,A2);
    if Sec < 0 %Si les Y tenen signe diferent apliquem Newton
        N=Newton_MS(A1,A2);
        Ntotal=Ntotal+1;
        B(Ntotal+1,:)=N;
    end
    A1=A2;
    x0=[A1(2),A1(3),A1(4),A1(5)];
    Ti=Tf;
    Tf=Tf+ht;
end
end

```

### A.7.5 TallsT\_TS

```
function B=TallsT_TS(TempsFi,ht,x0,tol)
%Funció que a partir d'un rang de temps que comença a 0 fins a TempsFi
%(explorant en salts de ht, és a dir l'increment de temps entre un
punt i el següent és ht)
%explora la òrbita que comença en el punt x0 fins que en troba els
talls a
%l'eix y=0 amb una tolerància tol_1.
%Per a poder executar-lo es necessiten els documents "FUN.m",
"Newton.m".
%Definició del temps
Ti=0;
Tf=-ht;
tspan=[Ti,Tf];
Ntotal=0;
B_1=0;
B_2=x0;
B(1,:)= [B_1,B_2];
options=odeset('MaxStep',ht,'RelTol',tol);
A1=[0,x0];
while Tf>=TempsFi
    %Algorisme de càlcul
    tspan=[Ti,Tf];
    [T,X]= ode45('FUN_T',tspan,x0,options);
    %Donem un nom més fàcil a la matriu per a poder treballar-la
    A_1=[T,X];
    [filesA_1,columnesA_1]=size(A_1);
    %D'aquesta manera A serà un vector que contindrà el següent
punt de la iteració
    A2=A_1(filesA_1,:);
    Sec=SeccioTS(A1,A2);
    if Sec < 0 %Si les Y tenen signe diferent apliquem Newton
        N=Newton_TS(A1,A2);
        Ntotal=Ntotal+1;
        B(Ntotal+1,:)=N;
    end
    A1=A2;
    x0=[A1(2),A1(3),A1(4),A1(5)];
    Ti=Tf;
    Tf=Tf-ht;
end
end
```

## A.8 Exploracions amb origen a Mart

### A.8.1 Càlcul d'una trajectòria

```
%Càlcul orbital versió Mart: donada una posició i una velocitat
inicials calcular la
òrbita.
%Es necessiten els documents "CJ.m" "FUN.m" i "ode78.m".
%%Definim els paràmetres d'entrada
%Definició de mu (en cas de variar-la l'haurem de escriure de nou a
FUN)
mu=3.22775159768e-07;
%Definició del temps:
tspan=[0,500];
%Valors inicials
x1=input('Entrar el valor inicial de x1= ');
x2=input('Entrar el valor inicial de x2= ');
x3=input('Entrar el valor inicial de x3= ');
```

```

x4=input('Entrar el valor inicial de x4= ');
x0=[x1;x2;x3;x4];
%Tipus de ode entrada:
ode_fcn_format=0;
%Entrem el valor de la tolerància:
tol = 1.e-6;
%Volem que es guardin tots els passos? si:trace=1, no:trace=0
trace=0;
%Contador no se per a què serveix:
count=0;
%Pas(step) màxim:
hmax=0.1;
%%Iniciem l'algorisme
[T,X]= ode78('FUN_M',tspan,x0,ode_fcn_format,tol,trace,count,hmax);
%Mostrar els resultats
plot(X(:,1),X(:,2),'r')
[T,X];

```

#### A.8.2 Càlcul dels talls per una constant de Jacobi en la secció $x=0$

```

%Exploració Final MART
%Generem els punts inicials
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
Cj=input('Entrar el valor de la Constant de Jacobi: ');
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
ht=0.1;
tol=10^-6;
%Generem la matriu de Punts Inicials on cada fila correspon a un punt
PI=PI_M(Num,NumAngles,Cj);
[filesPI,columnesPI]=size(PI);
contador=1;
disp('Inici Algoritme');
while contador<=filesPI
    x0=PI(contador,:);
    B_1=TallsT_M(TempsFi,ht,x0,tol);
    if contador==1
        B=B_1;
    else
        B=[B;B_1];
    end
    contador=contador+1;
end
%Guardem els resultats en vectors columna
t=B(:,1);
x1=B(:,2);
x2=B(:,3);
x3=B(:,4);
x4=B(:,5);
%Convertim els vectors columna en files
t=t';
x1=x1';
x2=x2';
x3=x3';
x4=x4';
%Creem un document amb els resultats
[stat,struc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATS'];
NameFile = [PathFolder '/MatriuTallsM.txt'];

```

```

mkdir([PathCurrent], '/RESULTATS');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID, '%15s %15s %15s %15s %15s\n', 't', 'x1', 'x2', 'x3', 'x4');
fprintf(fileID, '%15.5f %15.5f %15.5f %15.5f %15.5f\n', t, x1, x2, x3, x4);
fclose(fileID);
disp('Fi Algoritme');

```

### A.8.3 Càlcul dels talls per les tres constants de Jacobi en la secció $x=0$

```

%Exploració Final MART_2 per un rang de CJ
%Generem els punts inicials
clear all
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
%La Constant de Jacobi inicial es 3
Cj1=3;
%La Constant de Jacobi final és la del punt d'equilibri L1
mu=3.22775159768e-07;
Cj2=3+9*(mu/3)^(2/3)-7*(mu/3);
NumCj=input('Entrar el nombre de Cjs que volem explorar: ');
IncrementCj=(Cj2-Cj1)/(NumCj-1);
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
ht=0.1;
tol=10^-6;
disp('Inici Algoritme');
Cj=Cj1;
contadorCj=1;
while contadorCj<=NumCj
    text=['Explorem la Cj: ', num2str(Cj)];
    disp(text)
    %Generem la matriu de Punts Inicials on cada fila correspon a un
punt
    PIM=PI_M(Num, NumAngles, Cj);
    [filesPI, columnesPI]=size(PIM);
    contador=1;
    while contador<=filesPI
        x0=PIM(contador, :);
        if x0==[0,0,0,0]
            continue
        end
        B_1=TallsT_M(TempsFi, ht, x0, tol);
        %Generem el vector CjM que contindra el valor de la Cj
        [filesB_1, columnesB_1]=size(B_1);
        %No guardem el punt inicial
        if filesB_1==1
            break
        end
        contador2=1;
        clear CjM
        while contador2<=filesB_1
            CjM(contador2,1)=Cj;
            contador2=contador2+1;
        end
        B_2=[B_1(:,1), B_1(:,2), B_1(:,3), B_1(:,4), B_1(:,5), CjM(:,1)];
        if contador==1 && contadorCj==1
            B=B_2;
        else
            B=[B; B_2];
        end
    end
    Cj=Cj+IncrementCj;
    contadorCj=contadorCj+1;
end

```

```

end
    contador=contador+1;
end
Cj=Cj+IncrementCj;
contadorCj=contadorCj+1;
end
%Li donem un nom diferent per a poder comprar amb l'altre planeta
Mart=B;
%Creem un document amb els resultats
[stat,estruc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATSM'];
NameFile = [PathFolder '/MatriuTallsM.txt'];
mkdir([PathCurrent], '/RESULTATSM');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID,'%15s      %15s      %15s      %15s      %15s\n',
't','x1','x2','x3','x4','Cj');
fprintf(fileID,'%15.8f %15.8f %15.8f %15.8f %15.8f %15.8f\n',B');
fclose(fileID);
%Guardar la matriu del workspace
save([PathFolder '/BMart.mat'], 'Mart');
save([PathFolder '/PIMart.mat'], 'PIM');
disp('Fi Algoritme');

```

#### A.8.4 Càlcul dels talls per les tres constants de Jacobi en la secció $x=\mu$

```

%Exploració Final MART_3 per un rang de CJ en aquest cas la secció
serà
%la posició del Sol (x=-mu) enlloc de x=0
%Generem els punts inicials
clear all
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
%La Constant de Jacobi inicial es 3
Cj1=3;
%La Constant de Jacobi final és la del punt d'equilibri L1
mu=3.22775159768e-07;
Cj2=3+9*(mu/3)^(2/3)-7*(mu/3);
NumCj=input('Entrar el nombre de Cjs que volem explorar: ');
IncrementCj=(Cj2-Cj1)/(NumCj-1);
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
ht=0.1;
tol=10^-6;
disp('Inici Algoritme');
Cj=Cj1;
contadorCj=1;
while contadorCj<=NumCj
    text=['Explore la Cj: ',num2str(Cj)];
    disp(text)
    %Generem la matriu de Punts Inicials on cada fila correspon a un
punt
    PIM=PI_M(Num,NumAngles,Cj);
    [filesPI,columnesPI]=size(PIM);
    contador=1;
    if contadorCj==1
        PIM_2=PIM;
    else
        PIM_2=[PIM_2;PIM];
    end
end

```

```

while contador<=filesPI
    %Afegim un indicador del % en el que es troba el programa
    Percentatge=((contador-1)/filesPI)*100;
    text=['Cj   num:   ',num2str(contadorCj),'   de   ',num2str(NumCj),'
explorat un: ',num2str(Percentatge),' %'];
    disp(text)
    %Seguim amb el programa
    x0=PIM(contador,:);
    if x0==[0,0,0,0]
        continue
    end
    B_1=TallsT_MS(TempsFi,ht,x0,tol);
    %Generem el vector CjM que contindrà el valor de la Cj
    [filesB_1,columnesB_1]=size(B_1);
    contador2=1;
    clear CjM
    while contador2<=filesB_1
        CjM(contador2,1)=Cj;
        contador2=contador2+1;
    end
    B_2=[B_1(:,1),B_1(:,2),B_1(:,3),B_1(:,4),B_1(:,5),CjM(:,1)];
    if contador==1 && contadorCj==1
        B=B_2;
    else
        B=[B;B_2];
    end
    contador=contador+1;
end
Cj=Cj+IncrementCj;
contadorCj=contadorCj+1;
end
%Li donem un nom diferent per a poder comprar amb l'altre planeta
Mart=B;
%Creem un document amb els resultats
[stat,struc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATSM'];
NameFile = [PathFolder '/MatriuTallsM.txt'];
mkdir([PathCurrent], '/RESULTATSM');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID,'%15s          %15s          %15s          %15s          %15s\n',
't','x1','x2','x3','x4','Cj');
fprintf(fileID,'%15.8f %15.8f %15.8f %15.8f %15.8f %15.8f\n',B');
fclose(fileID);
%Guardar la matriu del workspace
save([PathFolder '/BMart.mat'], 'Mart');
save([PathFolder '/PIMart.mat'], 'PIM_2');
disp('Fi Algoritme');

```

## A.9 Exploracions amb origen a la Terra

### A.9.1 Càlcul d'una trajectòria

```

%Càlcul orbital versió Terra: donada una posició i una velocitat
inicials calcular la
%òrbita.
%Es necessiten els documents "CJ.m" "FUN.m" i "ode78.m".
%%Definim els paràmetres d'entrada
%Definició de mu (en cas de variar-la l'haurem de escriure de nou a
FUN)
mu=3.00652690848e-06;

```

```

%Definició del temps:
tspan=[0,-500];
%Valors inicials
x1=input('Entrar el valor inicial de x1= ');
x2=input('Entrar el valor inicial de x2= ');
x3=input('Entrar el valor inicial de x3= ');
x4=input('Entrar el valor inicial de x4= ');
x0=[x1,x2,x3,x4];
%Tipus de ode entrada:
ode_fcn_format=0;
%Entrem el valor de la tolerància:
tol = 1.e-6;
%Volem que es guardin tots els passos? si:trace=1, no:trace=0
trace=0;
%Contador no se per a què serveix:
count=0;
%Pas(step) màxim:
hmax=0.1;
%%Iniciem l'algorisme
options=odeset('MaxStep',hmax,'RelTol',tol);
[T,X]=ode45('FUN_T',tspan,x0,options);
%Mostrar els resultats
plot(X(:,1),X(:,2),'r')
[T,X];

```

#### A.9.2 Càlcul dels talls per una constant de Jacobi en la secció $x=0$

```

%Exploració Final TERRA
%Generem els punts inicials
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
Cj=input('Entrar el valor de la Constant de Jacobi: ');
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
TempsFi=-abs(TempsFi);
ht=0.1;
tol=10^-6;
%Generem la matriu de Punts Inicials on cada fila correspon a un punt
D=0.06;
%Càlcul de Punts inicials
PI=PI_T(D,Num,NumAngles,Cj);
[filesPI,columnesPI]=size(PI);
contador=1;
disp('Inici Algoritme');
while contador<=filesPI
    x0=PI(contador,:);
    B_1=TallsT_T(TempsFi,ht,x0,tol);
    if contador==1
        B=B_1;
    else
        B=[B;B_1];
    end
    contador=contador+1;
end
%Guardem els resultats en vectors columna
t=B(:,1);
x1=B(:,2);
x2=B(:,3);
x3=B(:,4);
x4=B(:,5);
%Convertim els vectors columna en files

```

```

t=t';
x1=x1';
x2=x2';
x3=x3';
x4=x4';
%Creem un document amb els resultats
[stat,estruc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATS'];
NameFile = [PathFolder '/MatriuTallsT.txt'];
mkdir([PathCurrent], '/RESULTATS');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID,'%15s %15s %15s %15s %15s\n','t','x1','x2','x3','x4');
fprintf(fileID,'%15.5f %15.5f %15.5f %15.5f %15.5f\n',t,x1,x2,x3,x4);
fclose(fileID);
disp('Fi Algoritme');
%Afegit per trobar D
plot(B(:,2),B(:,3))

```

### A.9.3 Càlcul dels talls per les tres constants de Jacobi en la secció $x=0$

```

%Exploració Final TERRA per un rang de CJ
clear all
%Generem els punts inicials
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
%La Constant de Jacobi inicial es 3
Cj1=3;
%La Constant de Jacobi final és la del punt d'equilibri L3
mu=3.00652690848e-06;
Cj2=3+2*mu-(49/48)*mu^2;
NumCj=input('Entrar el nombre de Cjs que volem explorar: ');
IncrementCj=(Cj2-Cj1)/NumCj;
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
TempsFi=-abs(TempsFi);
ht=0.1;
tol=10^-6;
D=0.06;
disp('Inici Algoritme');
Cj=Cj1;
contadorCj=1;
while contadorCj<=NumCj
    %Generem la matriu de Punts Inicials on cada fila correspon a un
punt
    PI=PI_T(D,Num,NumAngles,Cj);
    [filesPI,columnesPI]=size(PI);
    contador=1;
    while contador<=filesPI
        x0=PI(contador,:);
        if x0==[0,0,0,0]
            continue
        end
        B_1=TallsT_T(TempsFi,ht,x0,tol);
        %Generem el vector CjM que contindrà el valor de la Cj
        [filesB_1,columnesB_1]=size(B_1);
        contador2=1;
        clear CjM
        while contador2<=filesB_1
            CjM(contador2,1)=Cj;

```



```

        contador2=contador2+1;
    end
    B_2=[B_1(:,1),B_1(:,2),B_1(:,3),B_1(:,4),B_1(:,5),CjM];
    if contador==1 && contadorCj==1
        B=B_2;
    else
        B=[B;B_2];
    end
    contador=contador+1;
end
Cj=Cj+IncrementCj;
contadorCj=contadorCj+1;
end
%Li donem un nom diferent per a poder comprar amb l'altre planeta
Terra=B;
%Creem un document amb els resultats
[stat,struc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATST'];
NameFile = [PathFolder '/MatriuTallsT.txt'];
mkdir([PathCurrent], '/RESULTATST');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID,'%15s %15s %15s %15s %15s\n', 't', 'x1', 'x2', 'x3', 'x4', 'Cj');
fprintf(fileID,'%15.5f %15.5f %15.5f %15.5f %15.5f %15.5f\n',B');
fclose(fileID);
%Guardar la matriu del workspace
save([PathFolder '/BTerra.mat'], 'Terra');
disp('Fi Algoritme');

```

#### A.9.4 Càlcul dels talls per les tres constants de Jacobi en la secció $x=\mu$

```

%Exploració Final TERRA per un rang de CJ per un rang de CJ en aquest
cas la secció serà
%la posició del Sol (x=-mu) enlloc de x=0
clear all
%Generem els punts inicials
Num=input('Entrar el número de Files i Columnes (Num x Num) de la
matriu de Punts Inicials: ');
NumAngles=input('Entrar el número d angles (del vector velocitat) que
volem explorar per a cada punt inicial: ');
%La Constant de Jacobi inicial es 3
Cj1=3;
%La Constant de Jacobi final és la del punt d'equilibri L3
mu=3.00652690848e-06;
Cj2=3+2*mu-(49/48)*mu^2;
NumCj=input('Entrar el nombre de Cjs que volem explorar: ');
IncrementCj=(Cj2-Cj1)/NumCj;
TempsFi=input('Entrar el valor màxim de Temps que explorarem: ');
TempsFi=-abs(TempsFi);
ht=0.01;
tol=10^-6;
D=0.06;
disp('Inici Algoritme');
Cj=Cj1;
contadorCj=1;
contadorini=1;
while contadorCj<=NumCj
    text=['Explore la Cj: ',num2str(Cj)];
    disp(text)

```

```

    %Generem la matriu de Punts Inicials on cada fila correspon a un
punt
    PIT=PI_T(D,Num,NumAngles,Cj);
    [filesPI,columnesPI]=size(PIT);
    contador=1;
    if contadorCj==1
        PIT_2=PIT;
    else
        PIT_2=[PIT_2;PIT];
    end
while contador<=filesPI
    %Afegim un indicador del % en el que es troba el programa
    Percentatge=((contador-1)/filesPI)*100;
    text=['Cj   num:   ',num2str(contadorCj),'   de   ',num2str(NumCj),'
explorat un: ',num2str(Percentatge),' %'];
    disp(text)
    %Seguim amb el programa
    x0=PIT(contador,:);
    if x0==[0,0,0,0]
        continue
    end
    B_1=TallsT_TS(TempsFi,ht,x0,tol);
    %Generem el vector CjM que contindrà el valor de la Cj
    [filesB_1,columnesB_1]=size(B_1);
    contador2=1;
    clear CjM
    while contador2<=filesB_1
        CjM(contador2,1)=Cj;
        contador2=contador2+1;
    end
    B_2=[B_1(:,1),B_1(:,2),B_1(:,3),B_1(:,4),B_1(:,5),CjM];
    if contador==1 && contadorCj==1
        B=B_2;
    else
        B=[B;B_2];
    end
    contador=contador+1;
end
Cj=Cj+IncrementCj;
contadorCj=contadorCj+1;
end
%Li donem un nom diferent per a poder comprar amb l'altre planeta
Terra=B;
%Creem un document amb els resultats
[stat,struc] = fileattrib;
PathCurrent = struc.Name;
PathFolder = [PathCurrent '/RESULTATST'];
NameFile = [PathFolder '/MatriuTallsT.txt'];
mkdir([PathCurrent], '/RESULTATST');
% crear un archivo .txt con datos
fileID = fopen(NameFile,'w');
fprintf(fileID,'%15s           %15s           %15s           %15s           %15s\n',
't','x1','x2','x3','x4','Cj');
fprintf(fileID,'%15.5f %15.5f %15.5f %15.5f %15.5f %15.5f\n',B');
fclose(fileID);
%Guardar la matriu del workspace
save([PathFolder '/BTerra.mat'], 'Terra');
save([PathFolder '/PITerra.mat'], 'PIT_2');
disp('Fi Algoritme');

```

## A.10 Anàlisis dels resultats

## A.10.1 Coincidències de posicions

```

%Comparació si hi ha punts que coincideixin en posició
%Precisió de treball
Precisio=10^-6;
%Convertim la matriu de Mart en sistema de coordenades de la Terra
MartC=ConversioM(Mart);
%Comparació gràfica de les Y
[M,T]=Grafics(MartC,Terra);
%%Comparació analítica de les Y:
%Primera part: genera una matriu Mf i una Tf on hi ha els punts
coincidents.
[filesM,columnesM]=size(M);
[filesT,columnesT]=size(T);
contadorM=1;
contadorB=1;
NumCoincidencia=0;
while contadorM<=filesM
    contadorT=1;
    percentatge=100*contadorM/filesM;
    text=['Recerca de coincidències: ',num2str(percentatge),' %'];
    disp(text)
    while contadorT<=filesT
        if abs(M(contadorM,3)-T(contadorT,3))<=Precisio
            NumCoincidencia=NumCoincidencia+1;
            text=['Coincidència          trobada,          total:
',num2str(NumCoincidencia)];
            disp(text)
            T_1=T(contadorT,:);
            M_1=M(contadorM,:);
            if contadorB==1
                Tf=T_1;
                Mf=M_1;
            else
                Tf=[Tf;T_1];
                Mf=[Mf;M_1];
            end
            contadorB=contadorB+1;
        end
        contadorT=contadorT+1;
    end
    contadorM=contadorM+1;
end
%Segona part: genera una matriu OM i OT on troba l'origen dels punts
coincidents
[filesMf,columnesMf]=size(Mf);
[filesTf,columnesTf]=size(Tf);
[filesMart,columnesMart]=size(MartC);
[filesTerra,columnesTerra]=size(Terra);
contadorMf=1;
contadorTf=1;
contadorTerra=1;
contador2=1;
while contadorMf<=filesMf
    contadorMart=1;
    percentatge2=contadorMf/filesMf*100;
    text=['Explorat un ', num2str(percentatge2), '% de la matriu
MartC'];
    disp(text)

```

```

while contadorMart<=filesMart
    if MartC(contadorMart,1)==Mf(contadorMf,1)
        contador1=1;
        Estat=0;
        while Estat < 1
            if MartC(contadorMart-contador1,1)==0
                OM_1=MartC(contadorMart-contador1,:);
                if contador2==1
                    OM=OM_1;
                    contador2=contador2+1;
                else
                    OM=[OM;OM_1];
                end
                Estat=1;
            else
                contador1=contador1+1;
            end
        end
        contadorMart=contadorMart+1;
    end
    contadorMf=contadorMf+1;
end
contador2=1;
while contadorTf<=filesTf
    contadorTerra=1;
    percentatge2=contadorTf/filesTf*100;
    text=['Explorat un ', num2str(percentatge2), '% de la matriu
Terra'];
    disp(text)
    while contadorTerra<=filesTerra
        if Terra(contadorTerra,1)==Tf(contadorTf,1)
            contador1=1;
            Estat=0;
            while Estat < 1
                if Terra(contadorTerra-contador1,1)==0
                    OT_1=Terra(contadorTerra-contador1,:);
                    if contador2==1
                        OT=OT_1;
                        contador2=contador2+1;
                    else
                        OT=[OT;OT_1];
                    end
                    Estat=1;
                else
                    contador1=contador1+1;
                end
            end
            contadorTerra=contadorTerra+1;
        end
        contadorTf=contadorTf+1;
    end
end
disp('Els punts coincidents es troben en les matrius Mf i Tf')
disp('L origen dels punts coincidents es troben en les matrius OM i
OT')

```

#### A.10.2 Coincidències de posicions i velocitats

```

%Comparació si hi ha punts que coincideixin en posició en una
Precisio1 i
%en velocitat vectorialment en Precisio2

```

```

Precisio1=10^-6;
Precisio2=10^-3;
disp('Inici programa')
MartC=ConversioM(Mart);
disp('Martiu de Mart convertida')
%Comparació gràfica de les Y
[M,T]=Grafics(MartC,Terra);
%%Comparació analítica de les Y:
%Primera part: genera una matriu Mf i una Tf on hi ha els punts
coincidents.
[filesM,columnesM]=size(M);
[filesT,columnesT]=size(T);
contadorM=1;
contadorB=1;
NumCoincidencia=0;
while contadorM<=filesM
    contadorT=1;
    percentatge=100*contadorM/filesM;
    text=['Recerca de coincidències: ',num2str(percentatge),' %'];
    disp(text)
    while contadorT<=filesT
        if (abs(M(contadorM,3)-T(contadorT,3))<= Precisio1 &&
sqrt((M(contadorM,4)-T(contadorT,4))^2+(M(contadorM,4)-
T(contadorT,4))^2)<=Precisio2
            NumCoincidencia=NumCoincidencia+1;
            text=['Coincidència          trobada,          total:
',num2str(NumCoincidencia)];
            disp(text)
            T_1=T(contadorT,:);
            M_1=M(contadorM,:);
            if contadorB==1
                Tf=T_1;
                Mf=M_1;
            else
                Tf=[Tf;T_1];
                Mf=[Mf;M_1];
            end
            contadorB=contadorB+1;
        end
        contadorT=contadorT+1;
    end
    contadorM=contadorM+1;
end
%Segona part: genera una matriu OM i OT on troba l'origen dels punts
coincidents
if NumCoincidencia>0
    [filesMf,columnesMf]=size(Mf);
    [filesTf,columnesTf]=size(Tf);
    [filesMart,columnesMart]=size(MartC);
    [filesTerra,columnesTerra]=size(Terra);
    contadorMf=1;
    contadorTf=1;
    contadorTerra=1;
    contador2=1;
    while contadorMf<=filesMf
        contadorMart=1;
        percentatge2=contadorMf/filesMf*100;
        text=['Explorat un ', num2str(percentatge2), '% de la matriu
MartC'];
        disp(text)
        while contadorMart<=filesMart

```

```

    if MartC(contadorMart,1)==Mf(contadorMf,1)
        contador1=1;
        Estat=0;
        while Estat < 1
            if MartC(contadorMart-contador1,1)==0
                OM_1=MartC(contadorMart-contador1,:);
                if contador2==1
                    OM=OM_1;
                    contador2=contador2+1;
                else
                    OM=[OM;OM_1];
                end
                Estat=1;
            else
                contador1=contador1+1;
            end
        end
        contadorMart=contadorMart+1;
    end
    contadorMf=contadorMf+1;
end
contador2=1;
while contadorTf<=filesTf
    contadorTerra=1;
    percentatge2=contadorTf/filesTf*100;
    text=['Explorat un ', num2str(percentatge2), '% de la matriu
Terra'];
    disp(text)
    while contadorTerra<=filesTerra
        if Terra(contadorTerra,1)==Tf(contadorTf,1)
            contador1=1;
            Estat=0;
            while Estat < 1
                if Terra(contadorTerra-contador1,1)==0
                    OT_1=Terra(contadorTerra-contador1,:);
                    if contador2==1
                        OT=OT_1;
                        contador2=contador2+1;
                    else
                        OT=[OT;OT_1];
                    end
                    Estat=1;
                else
                    contador1=contador1+1;
                end
            end
            contadorTerra=contadorTerra+1;
        end
        contadorTf=contadorTf+1;
    end
    disp('Els punts coincidents es troben en les matrius Mf i Tf')
    disp('L origen dels punts coincidents es troben en les matrius OM
i OT')
else
    disp('No hem trobat coincidències')
end

```

### A.10.3 Coincidències de posicions i comparació de velocitats

%Programa que busca coincidències en posició dels punts i d'aquests en

```

%compara les velocitats per a generar-ne un gràfic.
Precisio1=10^-6;
%Convertim la matriu de Mart en sistema de coordenades de la Terra
MartC=ConversioM(Mart);
%Comparació gràfica de les Y
[M,T]=Grafics_2(MartC,Terra);
%%Comparació analítica de les Y:
%Primera part: genera una matriu Mf i una Tf on hi ha els punts
coincidents.
[filesM,columnesM]=size(M);
[filesT,columnesT]=size(T);
contadorM=1;
contadorB=1;
NumCoincidencia=0;
while contadorM<=filesM
    contadorT=1;
    percentatge=100*contadorM/filesM;
    text=['Recerca de coincidències: ',num2str(percentatge),' %'];
    disp(text)
    while contadorT<=filesT
        if abs(M(contadorM,3)-T(contadorT,3))<=Precisio1
            NumCoincidencia=NumCoincidencia+1;
            text=['Coincidència          trobada,          total:
',num2str(NumCoincidencia)];
            disp(text)
            T_1=T(contadorT,:);
            M_1=M(contadorM,:);
            if contadorB==1
                Tf=T_1;
                Mf=M_1;
            else
                Tf=[Tf;T_1];
                Mf=[Mf;M_1];
            end
            contadorB=contadorB+1;
        end
        contadorT=contadorT+1;
    end
    contadorM=contadorM+1;
end
%Segona part: genera una matriu OM i OT on troba l'origen dels punts
coincidents
[filesMf,columnesMf]=size(Mf);
[filesTf,columnesTf]=size(Tf);
[filesMart,columnesMart]=size(MartC);
[filesTerra,columnesTerra]=size(Terra);
contadorMf=1;
contadorTf=1;
contadorTerra=1;
contador2=1;
while contadorMf<=filesMf
    contadorMart=1;
    percentatge2=contadorMf/filesMf*100;
    text=['Explorat un ', num2str(percentatge2), '% de la matriu
MartC'];
    disp(text)
    while contadorMart<=filesMart
        if MartC(contadorMart,1)==Mf(contadorMf,1)
            contador1=1;
            Estat=0;
            while Estat < 1

```

```

        if MartC(contadorMart-contador1,1)==0
            OM_1=MartC(contadorMart-contador1,:);
            if contador2==1
                OM=OM_1;
                contador2=contador2+1;
            else
                OM=[OM;OM_1];
            end
            Estat=1;
        else
            contador1=contador1+1;
        end
    end
    end
    contadorMart=contadorMart+1;
end
contadorMf=contadorMf+1;
end
contador2=1;
while contadorTf<=filesTf
    contadorTerra=1;
    percentatge2=contadorTf/filesTf*100;
    text=['Explorat un ', num2str(percentatge2), '% de la matriu
Terra'];
    disp(text)
    while contadorTerra<=filesTerra
        if Terra(contadorTerra,1)==Tf(contadorTf,1)
            contador1=1;
            Estat=0;
            while Estat < 1
                if Terra(contadorTerra-contador1,1)==0
                    OT_1=Terra(contadorTerra-contador1,:);
                    if contador2==1
                        OT=OT_1;
                        contador2=contador2+1;
                    else
                        OT=[OT;OT_1];
                    end
                    Estat=1;
                else
                    contador1=contador1+1;
                end
            end
        end
        contadorTerra=contadorTerra+1;
    end
    contadorTf=contadorTf+1;
end
disp('Els punts coincidents es troben en les matrius Mf i Tf')
disp('L origen dels punts coincidents es troben en les matrius OM i
OT')
%Càlcul diferència de velocitat i gràfic
contador1=1;
while contador1<=filesTf
    DiferenciaVelocitat=sqrt((Mf(contador1,4)-
Tf(contador1,4))^2+(Mf(contador1,4)-Tf(contador1,4))^2)
    if contador1==1
        DifV=DiferenciaVelocitat;
    else
        DifV=[DifV;DiferenciaVelocitat];
    end
end

```



```

        contador1=contador1+1;
    end
    plot(Mf(:,3),DifV(:, 'r*'))
    title('Comparació velocitats')
    xlabel('Y mart')
    ylabel('Diferència velocitat')

```

#### A.10.4 Grafics

```

function [M,T]=Grafics (MartC,Terra)
%Funció que grafica per a 3 Cj diferents els talls de Mart i la Terra
en
%X=0
%Eliminem els punts inicials
contadorT1=1;
contadorT2=1;
[filesT,columnesT]=size(Terra);
while contadorT1<=filesT
    if Terra(contadorT1,1)==0
    else
        T(contadorT2,:)=Terra(contadorT1,:);
        contadorT2=contadorT2+1;
    end
    contadorT1=contadorT1+1;
end
contadorM1=1;
contadorM2=1;
[filesM,columnesM]=size(MartC);
while contadorM1<=filesM
    if MartC(contadorM1,1)==0
    else
        M(contadorM2,:)=MartC(contadorM1,:);
        contadorM2=contadorM2+1;
    end
    contadorM1=contadorM1+1;
end
%Mu terra
mu=3.00652690848e-06;
hold on
plot(T(:,2),T(:,3), 'o');
plot(M(:,2),M(:,3), 'r*');
end

```

#### A.10.5 Grafics\_2

```

function [M,T]=Grafics_2 (MartC,Terra)
%Funció que elimina els punts inicials
%X=0
%Eliminem els punts inicials
contadorT1=1;
contadorT2=1;
[filesT,columnesT]=size(Terra);
while contadorT1<=filesT
    if Terra(contadorT1,1)==0
    else
        T(contadorT2,:)=Terra(contadorT1,:);
        contadorT2=contadorT2+1;
    end
    contadorT1=contadorT1+1;
end
contadorM1=1;
contadorM2=1;
[filesM,columnesM]=size(MartC);

```

```

while contadorM1<=filesM
    if MartC(contadorM1,1)==0
    else
        M(contadorM2,:)=MartC(contadorM1,:);
        contadorM2=contadorM2+1;
    end
    contadorM1=contadorM1+1;
end
end

```

#### A.10.6 PlambOrbites

```

function PI=PIambOribtes(Mart)
%Funció que agafa els resultats de les ExploracionsFinales i ens dóna
els
%punts inicials que han tingut orbites amb talls
[filesMart,columnesMart]=size(Mart);
n=1;
m=1;
while n<filesMart
    if Mart(n,1)==0 && Mart(n+1,1)~=0
        PI_1=Mart(n,:);
        if m==1
            PI=PI_1;
        else PI=[PI;PI_1];
        end
        m=m+1;
    end
    n=n+1;
end
if Mart(filesMart,1)==0
    PI_1=Mart(filesMart,:);
    PI=[PI;PI_1];
end
end
end

```

#### A.10.7 ConversioM

```

function MartC=ConversioM(Mart)
%Funció que converteix la Matriu resposta de ExploracioFinalMart_2 en
el
%sistema de coordenades de la Terra
Conv=(227.9*10^6)/(149.6*10^6);
ConvT=1.880791328;
B1=ConvT*Mart(:,1);
B2=3.00652690848e-06 - Conv*(-Mart(:,2)+3.22775159768e-07);
B3=Conv*Mart(:,3);
B4=3.00652690848e-06 - (Conv/ConvT)*(-Mart(:,4)+3.22775159768e-07);
B5=(Conv/ConvT)*Mart(:,5);
%La conversió de la CJ no te sentit per tant la deixarem igual
B6=Mart(:,6);
MartC=[B1,B2,B3,B4,B5,B6];
end

```

## B- Pressupost

El pressupost desglossat es troba a la taula següent:

	<b>Cost horari (€/h)</b>	<b>Hores</b>	<b>Import(€)</b>
Recerca bibliogràfica	25	35	875
Programació de codi	25	200	5.000
Elaboració de documents	25	30	750
Amortització d'equips informàtics	0,90	1.200	1.080
Amortització de software	0,50	1.000	500
		<b>Total</b>	8.205