Mètodes numèrics

Gina Gardeñes Farré (1638998) — 2n de grau en Matemàtiques
Isaac Pizarro Contreras (1636421) — Curs 2023 - 2024

PRÀCTICA 2

1. Introducció

En aquesta segona pràctica de l'assignatura de Mètodes Numèrics, ens enfoquem en la implementació i validació de dos mètodes numèrics per a la cerca de zeros de funcions: el mètode de la bisecció i el mètode de Newton. L'objectiu és desenvolupar i provar aquestes tècniques per a la seva aplicació en el Problema Restringit de Tres Cossos (RTBP) i les òrbites de Lyapunov, proporcionant eines útils per al Departament de Dinàmica del Vol (DDV) de l'Agència Espacial Catalana (AEC).

El RTBP descriu el moviment d'una partícula de massa negligible sota la influència gravitatòria de dos cossos principals, com ara la Terra i la Lluna. Mitjançant la implementació dels mètodes de bisecció i Newton en llenguatge C, busquem zeros de funcions associades a les condicions de velocitat de les trajectòries del RTBP. La validació es realitza a través d'una sèrie d'experiments que avaluen la precisió i eficiència dels mètodes, amb l'objectiu final d'aconseguir una òrbita de Lyapunov periòdica.

Aquesta pràctica no només reforça els conceptes teòrics apresos, sinó que també demostra l'aplicabilitat dels mètodes numèrics en la resolució de problemes reals.

2. Implementació del software

En aquest apartat es comentaran decisions rellevants per al millor funcionament del codi, però no es farà una descripció intensiva de tota la programació realitzada ja que aquesta ja ve descrita en el manual del software d'aquesta mateixa memòria i els comentaris dels fitxers de C.
A la funció `newton`, es va implementar una verificació de la derivada per evitar divisions per zero.
Una decisió important va ser l'ús de la estructura `struct lyap ref` per emmagatzemar els paràmetres necessaris en les funcions `proptraj_bis` i `proptraj_nwt`. Aquesta estructura conté les variables `c` i `isgn`, que són essencials per la propagació de les trajectòries.

Utilitzar una `struct` per passar aquests paràmetres té diversos avantatges:

- **Organització**: Agrupar les variables en una estructura clarifica que aquestes variables estan relacionades i han de ser utilitzades conjuntament.
- **Mantenibilitat**: Si en el futur es necessiten més paràmetres, es poden afegir fàcilment a l'estructura sense necessitat de modificar totes les funcions que la utilitzen.
- **Claredat**: Redueix la complexitat de les funcions, ja que el nombre de paràmetres que cal passar és menor.

Tot i que podríem haver utilitzat variables globals per simplificar el pas de paràmetres, aquesta pràctica pot portar a problemes de mantenibilitat i errors difícils de depurar, especialment en programes més grans. Les variables globals poden ser modificades per qualsevol part del programa, fent que el codi sigui menys robust i més difícil de comprendre.

3. Manual del software

3.1. La funció bisecció

Aquesta funció aplica el mètode de bisecció per trobar l'arrel d'una funció dins d'un interval [a, b] amb una precisió especificada per `tol`.

Prototipus

```
double biseccio (double *a, double *b, double tol, double (*f)(double, void*),
        void *prm, int ixrr);
```

Paràmetres

- `a`: Punter al valor inicial de l'interval a.
- `b`: Punter al valor inicial de l'interval b.
- `tol`: Tolerància per a la solució.
- `f`: Funció de la qual es vol trobar l'arrel.
- `prm`: Paràmetres addicionals per a la funció f.
- `ixrr`: Indicador per imprimir les iteracions.

3.2. La funció de Newton

Aquesta funció aplica el mètode de Newton per trobar l'arrel d'una funció donada una aproximació inicial.

Prototipus

```
int newton(double *x, double tolf, double tolx, int maxit,
           void (*fdf)(double, double*, double*, void*), void *prm, int ixrr);
```

Paràmetres

- `x`: Punter al valor inicial de l'aproximació de l'arrel.
- `tolf`: Tolerància per al valor de la funció.
- `tolx`: Tolerància per al canvi en x.
- `maxit`: Nombre màxim d'iteracions.
- `fdf`: Funció que calcula f(x) i f'(x).
- `prm`: Paràmetres addicionals per a la funció fdf.
- `ixrr`: Indicador per imprimir les iteracions.

3.3. rtbp lyap csig

Aquest programa aplica la funció `proptraj` per propagar una trajectòria del RTBP i detecta un canvi de signe de `u` com a funció de `x`, per tal de localitzar una arrel que doni lloc a una òrbita de Lyapunov plana.

Comanda per generar l'executable: `make rtbp_lyap_csig`

3.4. rtbp lyap ref

Aquest programa aplica els mètodes de bisecció i Newton per trobar una arrel de la funció `proptraj` amb els paràmetres especificats en la línia de comandament.

Comanda per generar l'executable: `make rtbp_lyap_ref`

3.4.1. La funció `proptraj_bis`

Aquesta funció crida la funció `proptraj` amb els paràmetres especificats i retorna el valor resultant per a la bisecció.

Prototipus

```
double proptraj_bis(double x, void *prm);
```

Paràmetres

- `x`: Valor actual de `x`.
- `prm`: Punter als paràmetres estructurats de `lyap_ref`.

3.4.2. La funció `proptraj_nwt`

Aquesta funció crida la funció `proptraj` amb els paràmetres especificats i emmagatzema els resultats en `u` i `du` per a la iteració de Newton.

Prototipus

```
void proptraj_nwt(double x, double *u, double *du, void *prm);
```

Paràmetres

- `x`: Valor actual de `x`.
- `u`: Punter per emmagatzemar el valor de la funció `proptraj`.
- `du`: Punter per emmagatzemar la derivada de la funció `proptraj`.
- `prm`: Punter als paràmetres estructurats de `lyap_ref`.

4. Part experimental

Exercici 2 Expliqueu com és comporta la funció **bisecció()** amb els paràmetres de la següent taula i justifiqueu perquè el seu comportament és correcte.

a	b	tol	ixrr
0.5	1	0.5	1
0.5	1	0.25	1
0.5	1	0.0125	1
0.5	1	0.0625	1
0.5	1	0.03125	1
0.5	1	0.015625	1

Taula 1: Taula enumerada dels valors pendents d'estudi de la funció **bisecció()**

En aquesta part de la pràctica cal aplicar el mètode de bisecció a la funció $f(x) = e^x - 2$, en l'interval I=[a,b]=[0.5,1]. El mètode de bisecció és un algorisme que tracta de trobar l'arrel de la funció en un interval tancat aplicant el teorema de Bolzano per tal d'anar reduïnt a cada iterat l'amplada de l'interval on està localitzada l'arrel.
Com coneixem la solució de l'arrel de la funció amb la que estem treballant, $\alpha = \log 2 \approx 0{,}6931471806$, per tal de justificar el correcte comportament de la funció simplement compararem els valors entrats amb calculadora amb els donats per l'algorisme.
L'algorisme té els següents passos: per començar, triem el valor intermig de l'interval [0.5,1] i en calculem la seva imatge. A mesura que la tolerància és redueix, com veiem que el valor donat no s'apropa prou a $f(\alpha) = 0$, repetim l'acció però ara reestablint l'interval en el que treballem desplaçant-nos la mitja distància calculada abans i tornant a agafar el punt mig. Repetim el procés successivament fins que el valor obtingut s'aproxima òptimament al valor desitgat.
Un dels criteris que cal tenir en compte en el funcionament de l'algorisme és pel que fa als extrems dels intervals que va agafant, doncs a mesura que la funció va iterant i alhora l'interval va reduint en amplada, cap dels dos extrems es manté fix, sinó que tan un com l'altre van variant durant l'ejecució, això ve determinat pel signe de $f(m_i)$, és a dir, el signe que ens dona l'arrel aproximada al avaluar-la a funció. Si ho volem explicar com en el codi, aleshores el signe ve determinat per:

$$f(a_i)f(m_i) \leq 0$$

on $a = a_i$, $b = b_i$ i $m_i = \frac{b_i + a_i}{2}$. Així, si això passa, tenim que $a_{i+1} = a_i$ i $b_{i+1} = p_i$ i si no passa, ens queda $a_{i+1} = p_i$ i $b_{i+1} = b_i$.

És a dir, si el signe de la imatge no canvia, l'extrem a desplaçar serà el de la dreta (b), i si el signe sí canvia, aleshores l'extrem que es desplaçarà cap endins serà el de l'esquerra (a).
Una observació que hem de tenir en compte a l'hora d'estudiar el funcionament del codi és l'índex de tolerància. Aquest és l'únic valor de la taula que va canviant, i és el límit que s'interposa entre la distància de l'arrel al valor aproximat d'aquesta, per tant, és imaginable pensar que per a cada valor de la taula la funció bisecció simplement ens anirà afegint més iterats als ja obtinguts, això és degut a que cada cop la tolerancia va disminuïnt $\frac{b-a}{2^k} = \frac{1-0,5}{2^k}$ $\forall k = 1, 2, ..., 6$ respecte del valor incial, i per tant, és d'aquesta manera que l'exactitud de l'arrel aproximada es va fent cada cop més aguda i s'apropa més al resultat desitgat.

Si ens fixem un moment en el que estem fent, la tolerància tal i com l'hem definida, representa la desigualtat:

$$|m_k - \alpha| \leq \frac{b-a}{2^k}$$

on on m_k denoten les aproximacions de l'arrel $\alpha = \ln 2$ que anem trobant a cada pas. Així, té sentit considerar diferents aspectes sobre la desigualtat escrita:

- Ens diu que, $|m_k - \alpha| = |e(m_i, \alpha)| \leq \frac{b-a}{2^k}$. És a dir, que la tolerància és una fita superior de l'error absolut entre l'estimació de l'arrel i la pròpia arrel.
- En cas que α sigui l'única arrel de f a [a,b], demostra la convergència $\{m_i\}_{\text{í}}$ cap a α.
- Ens permet estimar el nombre d'iterats necessaris a l'algorisme de bisecció per a assolir certa precisió.

Ara doncs, estudiem el recorregut de la funció iterat a iterat. Per fer-ho, enregistrarem els resultats de la funció en una taula que indicarà com varien els intervals i quins valors de les arrels anem obtenint. Així, arribarem a veure que tot lo que hem explicat correspon tal i com volen a la pràctica.

- Toleràncies de 0.5, 0.25 i 0.125:

Iteració	**a**	**b**	**m**	**f(m)**
1	0.500000	1.000000	0.750000	0.1170000166126748

Taula 2: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.5.

Iteració	**a**	**b**	**m**	**f(m)**
1	0.500000	1.000000	0.750000	0.1170000166126748

Taula 3: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.25.

Iteració	**a**	**b**	**m**	**f(m)**
1	0.500000	1.000000	0.750000	0.1170000166126748

Taula 4: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.125.

Ens aquests tres primers casos, veiem que els iterats no varien, això és degut a que el mateix valor, $\alpha = 0{,}75$, entra dins de tots tres índexs de tolerància.

- Tolerància de 0.0625:

Iteració	**a**	**b**	**m**	**f(m)**
1	0.500000	1.000000	0.750000	0.1170000166126748
2	0.500000	1.000000	0.625000	-0.1317540425677777

Taula 5: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.0625.

L'arrel aproximada és de $\alpha = 0.625$.

- Tolerància de 0.0135:

Iteració	a	b	m	f(m)
1	0.500000	1.000000	0.750000	0.1170000166126748
2	0.500000	1.000000	0.625000	-0.1317540425677777
3	0.62500	0.750000	0.687500	-0.01126253041770808

Taula 6: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.01325.

L'arrel aproximada és de $\alpha = 0.687500$.

- Tolerància de 0.015625:

Iteració	a	b	m	f(m)
1	0.500000	1.000000	0.750000	0.1170000166126748
2	0.500000	1.000000	0.625000	-0.1317540425677777
3	0.62500	0.750000	0.687500	-0.01126253041770808
4	0.687500	0.750000	0.718750	0.05186677348797675

Taula 7: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.015625.

L'arrel aproximada és de α= 0.71875

- Tolerancia de 0.00015625:

Iteració	a	b	m	f(m)
1	0.500000	1.000000	0.750000	0.1170000166126748
2	0.500000	1.000000	0.625000	-0.1317540425677777
3	0.62500	0.750000	0.687500	-0.01126253041770808
4	0.687500	0.750000	0.718750	0.05186677348797675
5	0.687500	0.718750	0.703125	0.02005552770869645
6	0.687500	0.703125	0.695312	0.004335330874331245
7	0.687500	0.695312	0.691406	-0.003478832038737778
8	0.691406	0.695312	0.693359	0.0004244339097745353

Taula 8: Taula dels iterats de la funció **bisecció()** quan la tolerància és de 0.00015625

Tal i com es menciona en el guió de la pràctica, el mètode de bisecció no és del tot pràctic ni eficient a l'hora de trobar la solució. És un mètode lineal, i per tant, requereix d'un nombre elevat d'iterats per tal d'anar aproximant el valor en qüestió al desitjat. Això veiem que succeeix quan la tolerància cada cop és menor, i passa de requerir només una iteració a necessitar-ne vuit fins a d'aturar-se.

Exercici 4 Expliqueu com es comporta la funció **newton()** buscant un zero per a la funció $f(x) = e^x - 2$ que sabem que és $ln2$ amb els paràmetres de la següent taula i justifiqueu perquè aquest comportament és correcte:

	x	tolf	tolfx	maxit	ixrr
1	2	$1e^{-8}$	$1e^{-8}$	10	1
2	2	$1e^{-8}$	1	10	1
3	2	1	$1e^{-8}$	10	1
4	10	$1e^{-12}$	$1e^{-8}$	10	1
5	10	$1e^{-12}$	$1e^{-8}$	20	1

Taula 9: Taula enumerada amb els valors dels paràmetres pendents d'estudi de la funció **newton()**.

L'algorisme de Newton té més èxit i està més ben considerat per la seva eficiència que no pas l'algorisme de bisecció. Aquest mètode permet aproximar arrels d'equacions de la forma $f(x) = 0$. D'aquesta manera,

donada una funció $f : [a, b] \longrightarrow \mathbf{R}$ de classe ζ^2 i suposant que existeix $\alpha \in (a, b)$ amb $f(\alpha) = 0$, podem trobar una aproximació amb un algorisme de la forma:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

el qual és molt més ràpid sempre i quan $f'(\alpha) \neq 0$.
De fet, quan el procés d'aproximació s'inicia des d'una x_0 prou proper a l'arrel, aleshores a cada iterat aproximadament es duplica el seu nombre de decimals correctes, és per això que Newton és considerat un mètode quadràtic. En aquestes condicions, $|x_{i+1} - x_i| \approx |\alpha - x_i|$, per això la condició $|x_{i+1} - x_i| =\leq \epsilon$ és part del criteri d'aturada.

Tot i així, quan som lluny de l'arrel , el mètode de Newton pot convergir molt lentament o inclús no convergir, és per això que un requisit indispensable a l'hora d'aplicar aquest algorisme en funcionament és limitar el nombre d'iterats permesos, establint-hi un màxim.

Tot això que estem comentant ho podem veure en la taula que ens presenta l'enunciat. Per tant, nem a estudiar que passa en cada un dels casos particulars:

- Cas 1:

N^o **iteració**	**x**	**f(x)**	**f'(x)**
1	2	5.38905609893065	7.38905609893065
2	1.270670566473225	1.563241151464318	3.563241151464318
3	0.8319573037399686	0.2978118573744624	2.297811857374462
4	0.7023505840171669	0.01849176999946156	2.018491769999462
5	0.6931894022505122	8.444516382999367e-05	2.00008444516383
6	0.6931471814512683	$1{,}782646030790147e^{-09}$	2.000000001782646

Taula 10: Algorisme de Newton del cas 1 de la Taula 9.

Convergència assolida a la iteració 6: x = 0.6931471805599453, hem obtingut
x=0.6931471805599453 ; $\alpha = \ln(2)$=0.6931471805599453 i $|x - \alpha| = 0$

- Cas 2:

N^o **iteració**	**x**	**f(x)**	**f'(x)**
1	2	5.38905609893065	7.38905609893065
2	1.270670566473225	1.563241151464318	3.563241151464318
3	0.8319573037399686	0.2978118573744624	2.297811857374462
4	0.7023505840171669	0.01849176999946156	2.018491769999462
5	0.6931894022505122	8.444516382999367e-05	2.00008444516383
6	0.6931471814512683	$1{,}782646030790147e^{-09}$	2.000000001782646

Taula 11: Algorisme de Newton del cas 2 de la Taula 9.

Convergència assolida a la iteració 6: x = 0.6931471805599453, hem obtingut
x=0.6931471805599453 ; $\alpha = \ln(2) = 0.6931471805599453$ i $|x - \alpha| = 0$

- Cas 3:

N^o **iteració**	**x**	**f(x)**	**f'(x)**
1	2	5.38905609893065	7.38905609893065
2	1.270670566473225	1.563241151464318	3.563241151464318
3	0.8319573037399686	0.2978118573744624	2.297811857374462
4	0.7023505840171669	0.01849176999946156	2.018491769999462
5	0.6931894022505122	8.444516382999367e-05	2.00008444516383
6	0.6931471814512683	$1{,}782646030790147e^{-09}$	2.000000001782646

Taula 12: Algorisme de Newton del cas 3 de la Taula 9.

Convergència assolida a la iteració 6: x = 0.6931471805599453, hem obtingut
x=0.6931471805599453 ; $\alpha = \ln(2) = 0.6931471805599453$ i $|x - \alpha| = 0$

Ens aquest tres casos, veiem que el nombre d'iteracions i resultats són els mateixos, i és que l'únic paràmetre que canvia en cada un dels bucles són els índexs de tolerància de x_i i $f(x_i)$. Com que els bucles s'inicies des d'una x prou propera a l'arrel, té sentit que convergeixi en un nombre d'iterats més petit al establert.

- Cas 4:

N^o **iteració**	**x**	**f(x)**	**f'(x)**
1	10	22024.46579480672	22026.46579480672
2	9.000090799859525	8101.819719862136	8103.819719862136
2	8.000337597057529	2979.964519578422	2981.964519578422
4	7.001008295849171	1095.739446729389	1097.739446729389
5	6.002830221813441	402.5722037533379	404.5722037533379
6	5.007773715166285	147.5713767226605	149.5713767226605
7	4.021145257528353	53.76493448072802	55.76493448072802
8	3.05701008940924	19.26388490760056	21.26388490760056
9	2.151066279455919	6.594017137339266	8.594017137339266
10	1.383786318063988	1.989980397825394	3.989980397825394

Taula 13: Algorisme de Newton del cas 4 de la Taula 9.

No s'ha assolit la convergència després de 10 iteracions. x= 0.8850419134774479 ; $\alpha = \ln(2) = 0.6931471805599453$ $|\alpha - x| = 0.191895$

- Cas 5:

N^o **iteració**	**x**	**f(x)**	**f'(x)**
1	10	22024.46579480672	22026.46579480672
2	9.000090799859525	8101.819719862136	8103.819719862136
2	8.000337597057529	2979.964519578422	2981.964519578422
4	7.001008295849171	1095.739446729389	1097.739446729389
5	6.002830221813441	402.5722037533379	404.5722037533379
6	5.007773715166285	147.5713767226605	149.5713767226605
7	4.021145257528353	53.76493448072802	55.76493448072802
8	3.05701008940924	19.26388490760056	21.26388490760056
9	2.151066279455919	6.594017137339266	8.594017137339266
10	1.383786318063988	1.989980397825394	3.989980397825394
11	0.8850419134774479	0.4230859493155052	2.423085949315505
12	0.71043566418818	0.03487758885436021	2.03487758885436
13	0.6932957688719236	0.0002971987035365053	2.000297198703537
14	0.6931471915986418	2.207739324688873e-08	2.000000022077393
15	0.6931471805599453	0	2

Taula 14: Algorisme de Newton del cas 5 de la Taula 9.

Convergència assolida a la iteració 15: x = 0.6931471805599453

x= 0.6931471805599453 ; $\alpha = \ln(2) = 0.6931471805599453$ i $|\alpha - x| = 0$

I en aquests dos casos, on un dels paràmetres on cal recalcar el canvi és en la x_n amb la qual s'inicien els bucles, veiem que passa el que hem comentat anteriorment, que al tenir una x més allunyada de lo normal a l'arrel de la funció, el mètode comença el bucle molt lentament, i com es veu en el cas 4, amb només 10 iterats encara no arriba a convergir. Tot i així, en el cas 5, on es canvia el límit d'iteracions augmentant-lo fins a 20, veiem que finalment sí recupera la convergència i ràpidament troba l'aproximació que buscàvem. De fet, al principi, vieem que els canvis que precisa la x baixen d'un en un, això és perquè en x = 10, envia el primer iterat a $f(x) = 22024{,}46579480672$, i a partir d'aquí i fins que s'apropa a l'arrel:

$$x_{n+1} = x_n - \frac{e^{x_n} - 2}{e^{x_n}} = x_n - 1 + \frac{2}{e^{x_n}} \approx x_n - 1$$

Exercici 5 Representació de la gràfica de u com a funció de x corresponent a la memòria.

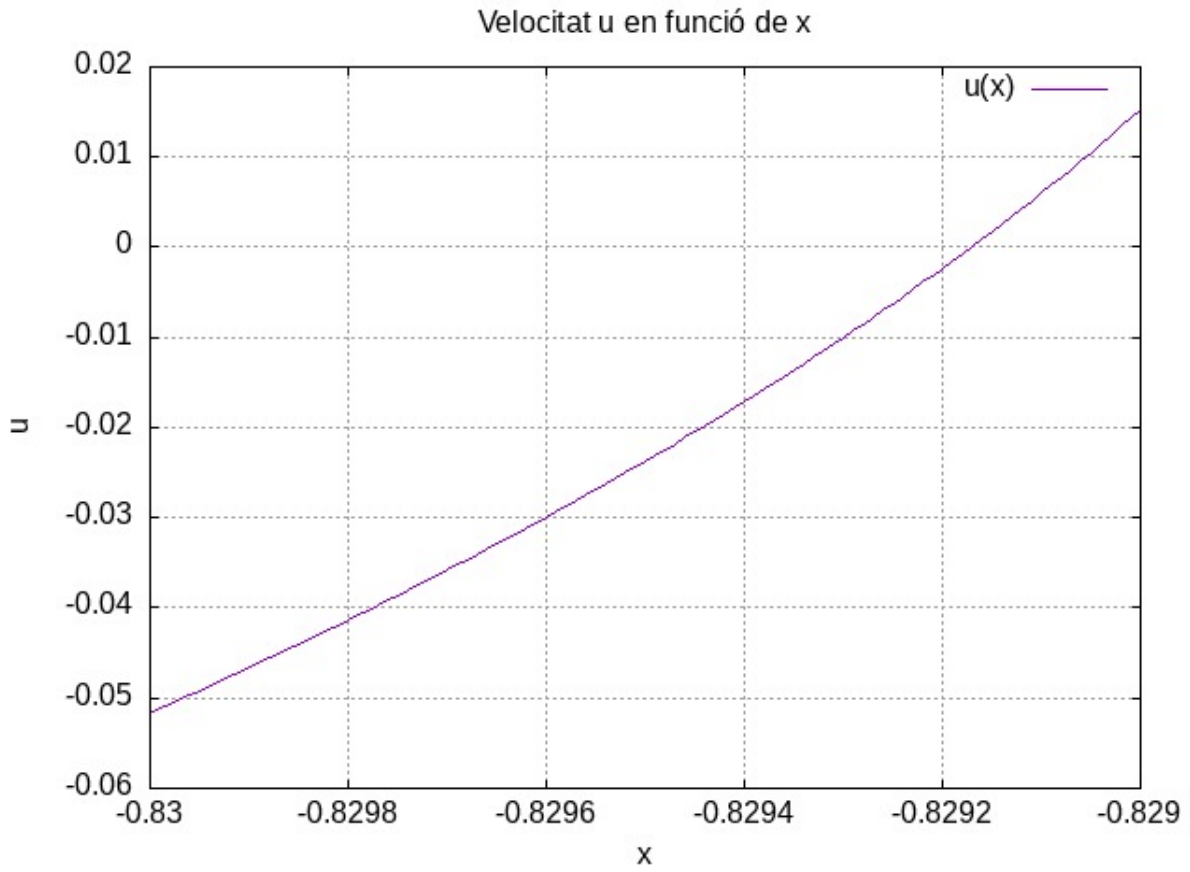

Figura 1: Gràfica de u com a funció de x.

Exerici 6 Representació de l'òrbita de Lyapunov plana.

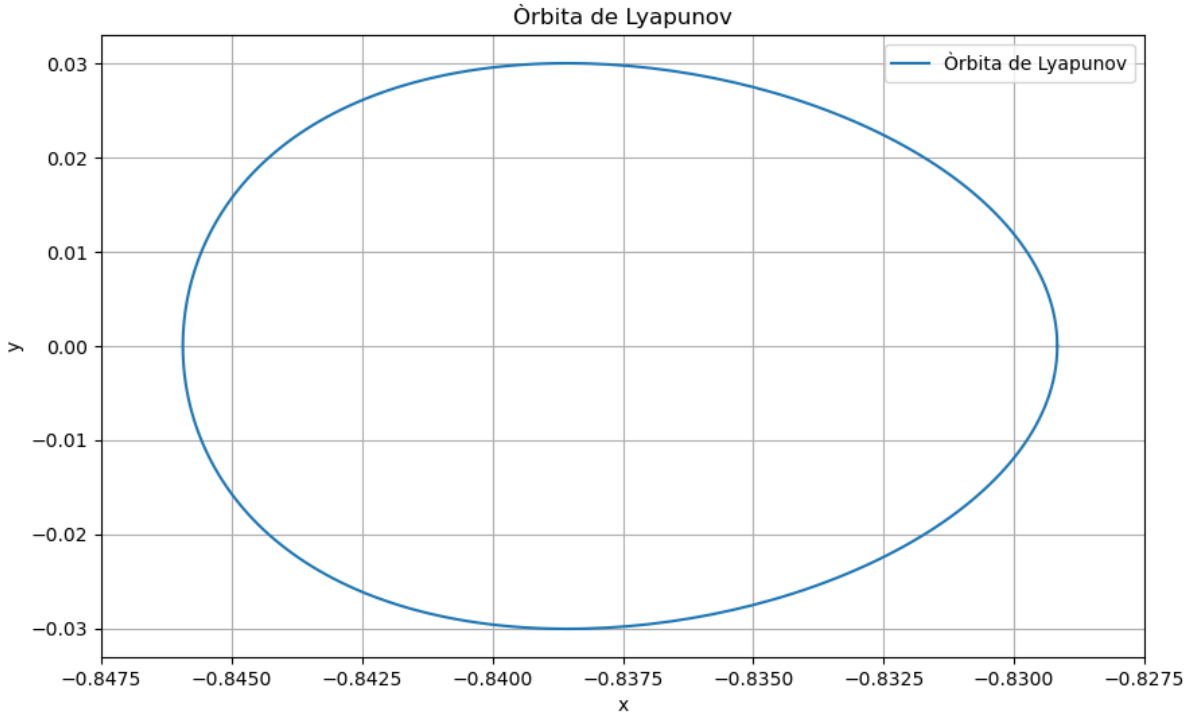

Figura 2: Òrbita de Lyapunov plana.

Aquesta òrbita, tal com es descriu en el guió de la pràctica, veiem que està tancada, és periòdica i per

tant, és de Lyapunov. A partir del setè tall, l'òrbita és trenc ja que són sistemes de treballs tan sensibles i fràgils que els propis càlculs d'arrodoniment dels ordenadors propaguen l'error i alteren els resultats que s'obtenen a mesura que anem iterant.

Exercici 7 Busqueu òrbites de Lyapunov addicionals de constant de Jacobi més baixa. Podeu trobar-ne alguna que s'apropi a la Lluna?

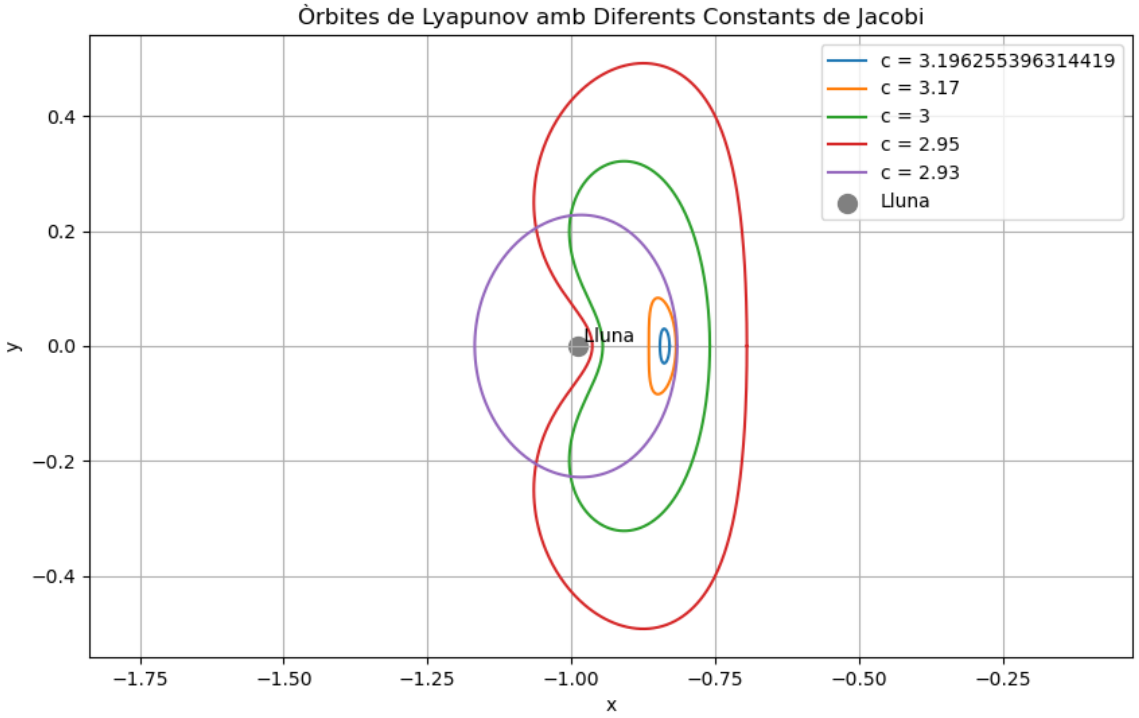

Figura 3: Òrbites de Lyapunov amb diferents Constants de Jacobi.