CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2014–2015
Pràctica en C

1 Objectius

Implementar el càlcul de dues maniobres impulsives per tal de canviar de certes posició i velocitat inicials a posició i velocitat finals prefixades, on el temps que separa les dues maniobres és un paràmetre donat.

2 Plantejament

Una diferència fonamental entre l'estudi de trajectòries de cossos celestes naturals (mecànica celeste) i l'estudi de trajectòries de cossos artificials (astrodinàmica) és que, en el cas de cossos artificials (naus tripulades, satèl·lits artificials, sondes interplanetàries), el fet de portar motors dóna la capacitat d'alterar la trajectòria mitjançant em maniobres. Això permet dissenyar trajectòries, que és la base del disseny de missions espacials.

Els dos tipus principals de propulsió que es fa servir a les missions espacials en l'actualitat són:[1]

- *Propulsió química*, en la qual s'apliquen impulsos grans de curta durada. Els motors en aquest cas són coets (grans o petits).
- *Propulsió iònica*, en la qual s'aplica un petit impuls de molt llarga durada. S'implementa amb motors elèctrics alimentats per panells solars.

Cada cop que s'engega el motor es diu que es fa una maniobra. En el primer cas, com que els motors estan engegats molt poc temps en relació a la durada de la missió, la nau es mou molt mentre els motors estan engegats poc en relació a la trajectòria total, i podem modelar la maniobra com un canvi instantani de velocitat (es parla de "fer una delta–v" ($\Delta \boldsymbol{v}$)). En el segon cas (propulsió iònica), s'ha d'afegir a les equacions diferencials un terme corresponent a l'impuls del motor, anomenat *terme de control*. L'estudi d'equacions diferencials amb termes de control des del punt de vista de com s'ha de triar el control per tal d'aconseguir una trajectòria desitjada forma part de la *teoria de control*.

En aquesta pràctica suposarem que emprem propulsió química, i per tant les maniobres es modelen com canvis instantanis de velocitat.

Per concretar, suposem que el moviment està governat per unes equacions diferencials d'aquest tipus:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad n = 2m, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{pmatrix}, \quad \boldsymbol{r}, \boldsymbol{v} \in \mathbb{R}^m. \tag{1}$$

L'espai de coordenades $\boldsymbol{r}$ s'anomena *espai de configuracions*, mentre que el de les coordenades $\boldsymbol{x} = (\boldsymbol{r}, \boldsymbol{v})^\top$ s'anomena *espai de fases*. Un exemple d'equacions d'aquest tipus són

[1] També s'està experimentant amb veles solars.

les equacions del pèndol:

$$\dot{r} = v, \quad \dot{v} = -\sin(r).$$

Denotarem $\Pi_{\boldsymbol{r}}\boldsymbol{x} = \boldsymbol{r}$, $\Pi_{\boldsymbol{v}}\boldsymbol{x} = \boldsymbol{v}$.

Un altre exemple són les equacions del *problema restringit de 3 cossos* en coordenades sinòdiques:

$$\dot{\boldsymbol{r}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = \boldsymbol{a}(\boldsymbol{r}, \boldsymbol{v}), \tag{2}$$

on, si denotem $\boldsymbol{r} = (x, y, z)^\top$, $\boldsymbol{v} = (u, v, w)^\top$,

$$\begin{aligned} \boldsymbol{a}(\boldsymbol{r}, \boldsymbol{v}) &= 2(v, -u, 0)^\top + \nabla\Omega(\boldsymbol{r}), \\ \Omega(\boldsymbol{r}) &= \frac{x^2 + y^2}{2} + \frac{1-\mu}{\rho_1} + \frac{\mu}{\rho_2} - \frac{1}{2}\mu(1-\mu), \\ \rho_1(\boldsymbol{r}) &= \left((x-\mu)^2 + y^2 + z^2\right)^{1/2}, \\ \rho_2(\boldsymbol{r}) &= \left((x-\mu+1)^2 + y^2 + z^2\right)^{1/2}. \end{aligned} \tag{3}$$

El problema restringit de 3 cossos descriu el moviment d'una sonda espacial sota l'atracció gravitatòria de 2 cossos (anomenats *primaris*) de masses $m_1 > m_2$ (com ara sol i terra, o terra i lluna) que se suposa que giren en cercles l'un al voltant de l'altre. $\boldsymbol{r}$ és el vector de posicions i $\boldsymbol{v}$ és el vector de velocitats, i estan referits a un sistema en el qual els dos cossos estan fixats a l'eix x, a les posicions $(\mu, 0, 0)^\top$ i $(\mu-1, 0, 0)^\top$, on $\mu = m_2/(m_1+m_2)$. Les unitats de temps són tals que els primaris completen una revolució en 2π unitats de temps. En el cas terra-lluna, 2π unitats de temps corresponen a un mes. En el cas terra-sol, corresponen a un any.

El nostre problema consistirà a: donats posicions i velocitats inicials $\boldsymbol{x}_0 = (\boldsymbol{r}_0, \boldsymbol{v}_0)^\top$, posicions i velocitats finals $\boldsymbol{x}_f = (\boldsymbol{r}_f, \boldsymbol{v}_f)^\top$ i un temps de vol Δt, trobar maniobres $\Delta\boldsymbol{v}_0$ i $\Delta\boldsymbol{v}_1$ tals que, partint de $(\boldsymbol{r}_0, \boldsymbol{v}_0)$, en aplicar $\Delta\boldsymbol{v}_0$, fer vol lliure durant Δt unitats de temps i aplicar $\Delta\boldsymbol{v}_1$, acabem amb posició i velocitat $(\boldsymbol{r}_f, \boldsymbol{v}_f)^\top$. Denotem el flux de les equacions diferencials (1) com

$$\boldsymbol{\phi}_t(\boldsymbol{x}_0),$$

que és l'única funció $t \mapsto \boldsymbol{\phi}_t(\boldsymbol{x}_0)$ que satisfà $\frac{d}{dt}\boldsymbol{\phi}_t(\boldsymbol{x}_0) = \boldsymbol{f}\big(\boldsymbol{\phi}_t(\boldsymbol{x}_0)\big)$ i $\boldsymbol{\phi}_0(\boldsymbol{x}_0) = \boldsymbol{x}_0$. El procés anterior el podem descriure com:

$$\begin{aligned} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 \end{pmatrix} &\xrightarrow{\text{mani } \Delta\boldsymbol{v}_0} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} \\ &\xrightarrow{\text{vol lliure}} \boldsymbol{\phi}_{\Delta t}\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} \\ &\xrightarrow{\text{mani } \Delta\boldsymbol{v}_1} \boldsymbol{\phi}_{\Delta t}\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix}. \end{aligned}$$

Per tant, volem trobar maniobres $(\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1) \in \mathbb{R}^n$ que anul·lin la funció

$$\widetilde{\boldsymbol{G}}\begin{pmatrix} \Delta\boldsymbol{v}_0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix} = \boldsymbol{\phi}_{\Delta t}\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix} - \begin{pmatrix} \boldsymbol{r}_f \\ \boldsymbol{v}_f \end{pmatrix}. \tag{4}$$

$\widetilde{\boldsymbol{G}}(\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1) = \boldsymbol{0}$ és un sistema no lineal de $n = 2m$ equacions. Com que les primeres m equacions no depenen de $\Delta\boldsymbol{v}_1$, es poden resoldre per separat buscant un zero de

$$\boldsymbol{G}(\Delta\boldsymbol{v}_0) := \Pi_{\boldsymbol{r}}\boldsymbol{\phi}_{\Delta t}\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} - \boldsymbol{r}_f = 0, \tag{5}$$

que és un sistema d'equacions no lineals $m \times m$. Un cop resolt, $\Delta\boldsymbol{v}_1$ s'obté aïllant de (4). De fet, (5) és la formulació del mètode del tir per resoldre un problema clàssic de valors a la frontera.

El sistema (5) es resol pel mètode de Newton. Per a això cal avaluar la diferencial de $\boldsymbol{G}$ (respecte de $\Delta\boldsymbol{v}_0$),

$$D\boldsymbol{G}(\Delta\boldsymbol{v}_0) = D_{\boldsymbol{v}}\big(\Pi_{\boldsymbol{r}}\boldsymbol{\phi}_{\Delta t}\big)\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix}, \tag{6}$$

que és una submatriu $m \times m$ de la diferencial del flux,

$$D\boldsymbol{\phi}_{\Delta t} = \left(\begin{array}{c|c} D_{\boldsymbol{r}}\big(\Pi_{\boldsymbol{r}}\boldsymbol{\phi}_{\Delta t}\big) & D_{\boldsymbol{v}}\big(\Pi_{\boldsymbol{r}}\boldsymbol{\phi}_{\Delta t}\big) \\ \hline D_{\boldsymbol{r}}\big(\Pi_{\boldsymbol{v}}\boldsymbol{\phi}_{\Delta t}\big) & D_{\boldsymbol{v}}\big(\Pi_{\boldsymbol{v}}\boldsymbol{\phi}_{\Delta t}\big) \end{array}\right),$$

que, a la seva vegada, és $A(\Delta t)$ on $A(t)$ és la solució del problema de valors inicials donat per l'EDO original amb les seves variacionals primeres acoblades,

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}), & \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \dot{A} &= D\boldsymbol{f}(\boldsymbol{x})A, & A(0) &= \mathrm{Id}\,. \end{aligned} \tag{7}$$

Denotem aquest problema de valors inicials per l'EDO ampliada com

$$\dot{\boldsymbol{X}} = \boldsymbol{F}(\boldsymbol{X}), \quad \boldsymbol{X}(0) = \boldsymbol{X}_0, \tag{8}$$

amb $\boldsymbol{X} \in \mathbb{R}^{n+n^2}$, $\boldsymbol{X} = (\boldsymbol{x}, A)$ amb A guardada **per columnes**. Denotem el flux d'aquest sistema ampliat com $\boldsymbol{\Phi}_t(\boldsymbol{X})$. Aleshores, si $(\boldsymbol{x}(t), A(t))$ és la solució del PVI (7), la relació entre les components de $\boldsymbol{x}(t), A(t)$ i les de $\boldsymbol{\Phi}_t(\boldsymbol{X}_0)$ ve donada per

$$\begin{aligned} \boldsymbol{x}(t) &= \Big(\Pi_i\boldsymbol{\Phi}_t(\boldsymbol{X}_0)\Big)_{i=0,\dots,n-1}, \\ A(t) &= \Big(\Pi_{n+jn+i}\boldsymbol{\Phi}_t(\boldsymbol{X}_0)\Big)_{i,j=0,\dots,n-1} \end{aligned}$$

Per a avaluar tant (5) com (6) cal integrar numèricament el sistema d'EDO (7) (en la forma (8)).

La integració numèrica d'EDO es planteja de manera més general per a sistemes d'EDO no autònoms. Tornem a fer servir $\boldsymbol{f}$ per denotar unes equacions diferencials ara no autònomes,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}), \quad \boldsymbol{x}(t_0) = x_0,$$

amb $\boldsymbol{x} \in \mathbb{R}^n$, i denotem el seu flux de temps t_0 a temps t per $\boldsymbol{\phi}_{t_0}^t(\boldsymbol{x})$, de manera que, fixats $t_0, \boldsymbol{x}_0$, la funció $t \mapsto \boldsymbol{\phi}_{t_0}^t(\boldsymbol{x})$ és l'única que satisfà

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\phi}_{t_0}^t(\boldsymbol{x}) = \boldsymbol{f}\big(t, \boldsymbol{\phi}_{t_0}^t(\boldsymbol{x})\big), \\ \quad \boldsymbol{\phi}_{t_0}^{t_0}(\boldsymbol{x}) = \boldsymbol{x}. \end{cases}$$

Una rutina d'integració numèrica d'un pas amb control de pas es pot fer servir com a "caixa negra" que, donades condicions inicials $(t_0, \boldsymbol{x}_0) \in \mathbb{R} \times \mathbb{R}^n$, un pas proposat h_0 i una tolerància tol > 0, torna $(t_1, \boldsymbol{x}_1) \in \mathbb{R} \times \mathbb{R}^n$ i un nou pas proposat h_1, satisfent:

- $\|\boldsymbol{\phi}_{t_0}^{t_1}(\boldsymbol{x}_0) - \boldsymbol{x}_1\| < \text{tol}$,
- t_1 es torna tant proper a $t_0 + h_0$ com es pugui, satisfent la condició anterior,
- h_1 és un nou pas proposat, amb el que s'espera que una propera aplicació del mètode permeti trobar $\boldsymbol{\phi}_{t_1}^{t_1+h_1}(\boldsymbol{x}_1)$ amb tolerància tol.

Teniu disponible una rutina `rk78()` que fa això. Com que el pas és variable, no es pot garantir que arribi a un t donat. Per això, per a poder avaluar el flux $\boldsymbol{\phi}_{t_0}^{t}(\boldsymbol{x}_0)$ heu d'escriure una altra rutina que anomenarem `flux()`, que cridi `rk78()` tantes vegades com calgui i ajusti el pas de la darrera crida per a arribar a temps t. Per comoditat (però sense perdre generalitat), com que tots els sistemes d'EDO amb què treballarem són autònoms, la rutina `flux()` que heu de fer ha d'avaluar $\boldsymbol{\phi}_{t_0}^{t_0+T}(\boldsymbol{x}_0)$ donats t_0 i T, en comptes de $\boldsymbol{\phi}_{t_0}^{t}(\boldsymbol{x}_0)$ donats t_0 i t,

3 Guió

1. Escriviu una rutina amb prototipus

```
int flux (
      double *t, double x[], double *h, double T,
      double pasmin, double pasmax, double tol, int npasmx,
      int n,
      int (*camp)(int n, double t, double x[], double f[], void *prm),
      void *prm
      );
```

   que avaluï $\boldsymbol{\phi}_{t_0}^{t_0+T}(\boldsymbol{x})$, on t_0 l'entreu a `*t`, $\boldsymbol{x}$ l'entreu a `x[0],...,x[n-1]`, i a `*h` entreu un pas proposat. A la sortida, aquesta rutina ha de tornar $t_0 + T$ a `*t`, $\boldsymbol{\phi}_{t_0}^{t_0+T}(\boldsymbol{x})$ a `x[]`, i a `*h` la proposta de pas per a crides posteriors. La resta de paràmetres és:

   - `pasmin`, `pasmax`, `tol` són resp. el pas mínim, pas màxim i la tolerància per passar a `rk78()`,
   - El paràmetre `npasmax` és el nombre màxim de crides a `rk78()` que permetem.
   - Els paràmetre `n` és el nombre de variables dependents (dimensió del sistema d'EDO), `camp` és un punter a funció que implementa les equacions que volem integrar, i `prm` són paràmetres per passar al camp (μ en el cas del RTBP, σ, ρ, β en el cas del model de Lorenz més avall).

2. Escriviu utilitats que generin punts equiespaiats en el temps al llarg de trajectòries del pèndol, del model de Lorenz i del RTBP. Concretament:

(a) Escriviu dins un fitxer `pendol_int.c` una utilitat que respongui a la crida

```
./pendol_int
```

(sense arguments a la línia de comandes). Per standard input li heu d'entrar línies de la forma

```
x1_0 x2_0 T np
```

Per cada línia d'aquesta forma, `./pendol` ha de bolcar `np+1` punts equiespaiats en el temps partint de $\boldsymbol{x}_0 = (\texttt{x1\_0}, \texttt{x2\_0})^\top$, cadascun d'ells a una línia de la forma

```
t  x1  x2
```

amb $\texttt{t} = jT/\texttt{np}$ i $(\texttt{x1}, \texttt{x2}) = \boldsymbol{\phi}_{jT/\texttt{np}}(\boldsymbol{x}_0)$ per a $j = 0, 1, \ldots, \texttt{np}$. Els blocs de punts corresponents a diferents trajectòries han d'estar separat per **dues** línies en blanc.

Per exemple, la crida

```
echo "0 0.3 7 100 0 0.6 7 100" | ./pendol > pendol.txt
```

ha de generar dues trajectòries de libració, que s'han de poder representar amb `gnuplot` mitjançant

```
set size ratio -1
plot 'pendol.txt' u 2:3 w l
```

Feu una figura del pèndol amb prou trajectòries com perquè es vegi el retrat-fase complet.

Nota: abans de tot això, heu d'escriure una rutina amb prototipus

```
int pendol (int n, double t, double x[], double f[], void *prm);
```

que implementi les equacions del pèndol. Aquesta rutina ha de ser dins un fitxer anomenat `pendol.c`. També heu d'escriure un fitxer de capçalera `pendol.h` amb el seu prototipus (que no és res més que la línia anterior).

(b) Escriviu una utilitat que respongui a la crida

```
./lorenz_int sgm rho bet
```

que generi trajectòries equiespaiades en el temps del model de Lorenz, que ve donat pel sistema d'equacions diferencials

$$\begin{cases} \dot{x}_1 &= \sigma(x_2 - x_1), \\ \dot{x}_2 &= \rho x_1 - x_2 - x_1 x_3, \\ \dot{x}_3 &= x_1 x_2 - \beta x_3. \end{cases}$$

Els arguments `sgm,rho,bet` de la línia de comandes de `./lorenz_int` són els paràmetres σ, ρ, β d'aquestes equacions diferencials. L'entrada per standard input ha de ser similar a l'anterior, però aquest cop amb tres coordenades. És a dir, línies de la forma

```
x1_0 x2_0 x3_0 T np
```

Feu una figura que representi l'*l'atractor de Lorenz*: el podeu generar amb la comanda

```
echo \
"0.416460744911 0.908936263452 0.0143831116293 150 25000" \
| ./lorenz_int 10=sgm 28=rho 2.66666666666667=bet > lorenz.txt
```

i representar-lo dins `gnuplot` amb

```
set term x11 size 700,700
set view equal xyz
set ticslevel 0.1
splot 'lorenz.txt' u 2:3:4 w l
```

Nota: com abans, haureu d'escriure una rutina que implementi les equacions diferencials anteriors dins un fitxer `lorenz.c`, i la corresponent capçalera a `lorenz.h`.

(c) Escriviu una utilitat similar per al RTBP, que respongui a la crida

```
./rtbps_int mu
```

on `mu` és el paràmetre μ a (3). Representeu la família d'òrbites halo al voltant de L_1 per al cas Terra-Lluna mitjançant la crida (a la línia de comandes Unix)

```
./rtbps_int 1.215058560962404e-2 < halos.inp > halos.txt
```

i les ordres gnuplot:

```
set term x11 size 700,700
set view equal xyz
set ticslevel 0.1
set ytics .1
set xlabel 'x' ; set ylabel 'y' ; set zlabel 'z'
splot 'halos.txt' u 2:3:4 w l,'L1.txt' w p,'trr.txt' w p,'lln.txt' w p
```

Depenent del vostre sistema operatiu, potser heu de canviar `x11` a la primera ordre gnuplot per `wxt`, `windows` o `aqua`.

3. Al fitxer `rtbps.c` trobareu una rutina amb prototipus

```
int rtbps (int n, double t, double *x, double *f, void *prm);
```

que implementa les equacions diferencials (2), (3) del problema restringit de tres cossos. No fa només això. Si se li entra `n` $= 6$, torna dins `f[]` les equacions diferencials en la forma (1). Si se li entra `n` $= 42$, torna dins `f[]` les equacions diferencials amb variacionals primeres acoblades en la forma (8).

Comproveu numèricament que $A(t)$ de (7) és igual a $D\boldsymbol{\phi}_t(\boldsymbol{x}_0)$. $A(t)$ s'avalua cridant `flux()` amb el camp `rtbps()` amb `n` $= 42$. Per altra banda, $D\boldsymbol{\phi}_t(\boldsymbol{x}_0)$ el podeu aproximar diferenciant numèricament $\boldsymbol{\phi}_t(\boldsymbol{x}_0)$, que s'avalua mitjançant crides a `flux()` amb el camp `rtbps()` amb `n` $= 6$.

4. Dins un fitxer `cmani.c`, escriviu una rutina amb prototipus

```
int cmani_gdg (int m, double x0[], double xf[], double dt, double dv[],
      double g[], double dg[], double pivfl[],
      double pas0, double pasmin, double pasmax, double tolfl, int npasmx,
      int (*camp)(int n, double t, double x[], double f[], void *prm),
      void *prm
      );
```

que avaluï tant $\boldsymbol{G}$ definida a (5) com $D\boldsymbol{G}$ definida a (6). A la llista d'arguments:

- `x0` (entrada) apunta a un vector de llargada `2*m` que conté $\Delta\boldsymbol{r}_0, \Delta\boldsymbol{v}_0$.
- `xf` (entrada) apunta a un vector de llargada `2*m` que conté $\Delta\boldsymbol{r}_f, \Delta\boldsymbol{v}_f$.
- `dt` (entrada) és Δt.
- `dv` (entrada) apunta a un vector de llargada `m` que conté $\Delta\boldsymbol{v}_0$.
- `g` (sortida) apunta a un vector de llargada `m` on s'ha de guardar $\boldsymbol{G}(\Delta\boldsymbol{v}_0)$ (eq. (5)).
- `dg` (sortida) apunta a un vector de llargada `m*m` on s'ha de guardar la matriu $D\boldsymbol{G}(\Delta\boldsymbol{v}_0)$ (6) per columnes.
- `pivfl` (sortida) apunta a un vector de llargada `m` on s'ha de guardar $\Pi_v \boldsymbol{\phi}_{\Delta t}\binom{\boldsymbol{r}_0}{\boldsymbol{v}_0+\Delta\boldsymbol{v}_0}$ ("projecció (pi) sobre v del flux").
- `pas0,pasmin,pasmax,tolfl,npasmx` (entrada) són per passar a `flux()`.
- `camp` (entrada) apunta a una funció que implementa el camp vectorial. Aquesta funció ha de fer com `rtbps()` respecte del seu primer argument `n`: si és `2*m` ha d'integrar les equacions diferencials (eq. (1)), mentre que si és `2*m*(1+2*m)` ha d'integrar les equacions diferencials amb variacionals acoblades (eq. (8)), amb la diferencial del flux guardada per columnes.

Aquesta funció ha de tornar 0 si tot ha anat bé, $\neq 0$ si s'ha produït algun error.

5. Dins el mateix fitxer anterior `cmani.c`, escriviu una rutina amb prototipus

```
int cmani (
      int m, double x0[], double xf[], double dt, double dv[],
      double tol, int maxit,
      double pas0, double pasmin, double pasmax, double tolfl, int npasmx,
      int (*camp)(int n, double t, double x[], double f[], void *prm),
      void *prm
      );
```

que trobi les maniobres $\Delta\boldsymbol{v}_0$, $\Delta\boldsymbol{v}_1$ que resolguin $\widetilde{\boldsymbol{G}}(\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1) = 0$, amb $\widetilde{\boldsymbol{G}}$ definida a (4). A la llista d'arguments:

- `m,x0,xf,dt,pas0,pasmin,pasmax,tolfl,npasmx,camp,prm` (entrada) són com a `cmani_gdg()`.
- `dv` (entrada/sortida) apunta a un vector de llargada `2*m` que conté $\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1$. A l'entrada són aproximacions inicials, i a la sortida han de ser les maniobres buscades, o sigui, solució de $\widetilde{\boldsymbol{G}}\binom{\Delta\boldsymbol{v}_0}{\Delta\boldsymbol{v}_1} = 0$ (eq. (4)).
- `tol,maxit` (entrada) són tolerància i màxim número d'iterats permesos per al mètode de Newton.

6. Comproveu les dues rutines anteriors resolent el següent problema per al pèndol: partint de $r_0 = 1$, $v_0 = 0$, volem trobar les maniobres Δv_0, Δv_1 per a arribar a $r_f = 0$ amb velocitat $v_f = -\sqrt{2(1-\cos 1)} = -0.9588\,5108$ (que és la corresponent a la trajectòria que passa per $(1, 0)$) en temps $\Delta t = \pi/2 = 1.5707\,9633$. Obtindreu com a resultat:

```
cmani(): it 0 ng 0.0997294 nc 0.0879976
cmani(): it 1 ng 0.00081458 nc 0.000730454
cmani(): it 2 ng 5.17021E-08 nc 4.63685E-08
cmani(): it 3 ng 4.56666E-16
dv[] -0.08872812680254195 0.004096513978830041
```

   Observeu la convergència quadràtica. Respecte del resultat: la trajectòria per (r_0, v_0) ja passa per (r_f, v_f), però no en el temps demanat. Triga més del temps demanat perquè el període decreix amb la mida de les oscil·lacions i té límit 2π. Per això cal accelerar (instantàniament) a la sortida i frenar (també instantàniament) a l'arribada.

   Preneu $\Delta\boldsymbol{v}_0 = \Delta\boldsymbol{v}_1 = 0$ com a aproximació inicial per a fer les iteracions de Newton.

7. Escriviu una utilitat `cmani_rtbp` que calculi dues maniobres impulsives al problema restringit de tres cossos per a anar dels estats $\boldsymbol{x}_0 = (\boldsymbol{r}_0, \boldsymbol{v}_0)^\top$ als estats $\boldsymbol{x}_1 = (\boldsymbol{r}_1, \boldsymbol{v}_1)^\top$. Aquesta utilitat ha de respondre a la crida

   ```
   ./cmani_rtbp mu tolnwt maxitnwt
   ```

   Ha de rebre línies de la forma

   ```
   dt x0[0] x0[1] x0[2] x0[3] x0[4] x0[5] xf[0] xf[1] xf[2] xf[3] xf[4] xf[5]
   ```

   i tornar línies de la forma

   ```
   dv[0] dv[1] dv[2] dv[3] dv[4] dv[5]
   ```

   on (`dv[0],dv[1],dv[2]`) i (`dv[3],dv[4],dv[5]`) són les components de la maniobres $\Delta\boldsymbol{v}_0$ i $\Delta\boldsymbol{v}_1$ (respectivament) que cal fer per a anar de $\boldsymbol{x}_0$ donat per `x0[]` a $\boldsymbol{x}_f$ donat per `xf[]`. Com abans, preneu $\Delta\boldsymbol{v}_0 = \Delta\boldsymbol{v}_1 = 0$ com a aproximació inicial per a fer les iteracions de Newton.

   Comproveu la utilitat amb aquest exemple: a partir de la crida

   ```
   ./cmani_rtbp 1.215058560962404e-2 1e-12=tolnwt < cmani_rtbp.inp
   ```

   heu d'obtenir com a sortida el contingut del fitxer `cmani_rtbp.out`.