**CÀLCUL NUMÈRIC**
Grau de Matemàtiques, 2022–2023
**Pràctica en C**

**Control discret de trajectòries mitjançant maniobres impulsives**[1]

**Resum**

Suposem donat un moviment a $\mathbb{R}^n$ amb $n = 2m$, on suposem que les primeres $m$ variables descriuen posicions i les darreres $m$ variables descriuen velocitats. L'objectiu d'aquesta pràctica és implementar el càlcul de dues maniobres impulsives (canvis instantanis de velocitat) per tal de, donades posicions i velocitats inicials i finals i un temps de vol, poder anar de les posicions i velocitats inicials a les posicions i velocitats finals en el temps de vol demanat.

# Introducció

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 (siguin naus tripulades, satèl·lits artificials o sondes interplanetàries), el fet de portar motors dóna la capacitat d'alterar la trajectòria mitjançant maniobres. Això permet que les trajectòries de les missions espacials es puguin dissenyar.

Sense tenir en compte les veles solars i altres tecnologies experimentals, els dos tipus principals de propulsió que es fan servir a les missions espacials en l'actualitat són:

- *Propulsió química*, en la qual s'apliquen forces grans de curta durada (es pot menysprear el desplaçament de la sonda durant l'estona en què els motors han estat engegats). Els motors en aquest cas són coets (grans o petits).

- *Propulsió iònica*, en la qual s'aplica una força petita de molt llarga durada (dies, setmanes, o fins i tot mesos). S'implementa amb motors elèctrics alimentats per panells solars.

En el primer cas, l'efecte que té sobre la trajectòria el fet d'engegar els motors es pot modelar com un canvi de velocitat instantani, o sigui, sense que tingui lloc un canvi de posició. Es diu que s'ha fet una maniobra, o, més en l'argot d'enginyers i controladors, "s'ha aplicat una delta-v" ($\Delta \boldsymbol{v}$). En el segon cas (propulsió iònica), aquesta simplificació no és realista i s'ha d'afegir a les equacions diferencials un terme corresponent a l'acceleració afegida pel 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. Aquests canvis instantanis de velocitat també es poden considerar com una forma de control discret. Quan s'insereixen moltes maniobres i es fa tendir el nombre de maniobres a infinit, es pot obtenir el control (continu) de la teoria de control com a límit.

[1] Pràctica dissenyada pel professor Josep Maria Mondelo

Suposarem 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 les equacions del pèndol:

$$\dot{r} = v, \quad \dot{v} = -\sin(r). \tag{2}$$

A la figura 1 trobareu el *retrat-fase* del pèndol, que és una representació esquemàtica de totes les seves trajectòries.

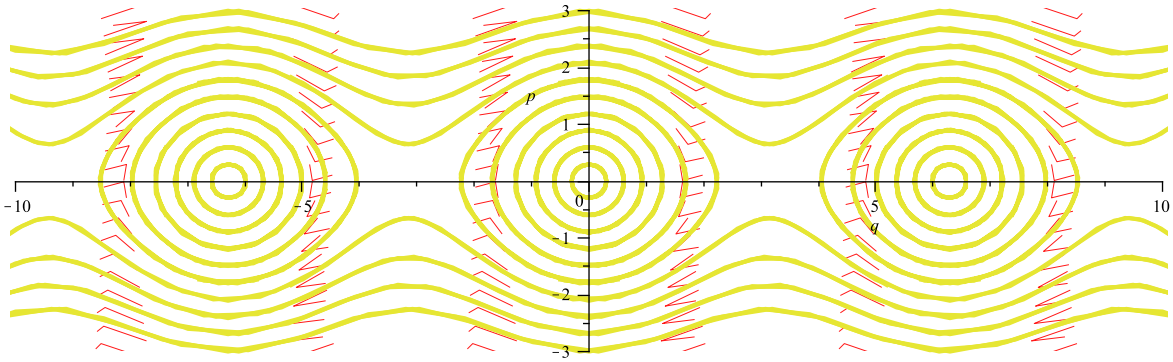


Figura 1: Retrat–fase del pèndol (eq. (2)).

Un altre exemple són les equacions del *problema restringit de 3 cossos* (RTBP, de les sigles en anglès) en coordenades sinòdiques:

$$\dot{\boldsymbol{r}} = \boldsymbol{v}, \quad \dot{\boldsymbol{v}} = \boldsymbol{a}(\boldsymbol{r}, \boldsymbol{v}), \tag{3}$$

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

$$\begin{aligned} \boldsymbol{a}(\boldsymbol{r}, \boldsymbol{v}) &= 2(v, -u, 0)^\top + \nabla\Omega(\boldsymbol{r}), \\ \Omega(\boldsymbol{r}) &= \frac{x^2 + y^2}{2} + \frac{1-\mu}{\rho_1} + \frac{\mu}{\rho_2} - \frac{1}{2}\mu(1-\mu), \\ \rho_1(\boldsymbol{r}) &= \left((x-\mu)^2 + y^2 + z^2\right)^{1/2}, \\ \rho_2(\boldsymbol{r}) &= \left((x-\mu+1)^2 + y^2 + z^2\right)^{1/2}. \end{aligned} \tag{4}$$

El RTBP descriu el moviment d'una sonda espacial sota l'atracció gravitatòria de 2 cossos (anomenats *primaris*) de masses $m_1 \geq m_2$ (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 $2\pi$ unitats de temps. En el cas terra-lluna, $2\pi$ unitats de temps corresponen a un mes. En el cas terra-sol, corresponen a un any.

En aquesta formulació, el RTBP té 5 punts d'equilibri anomenats *punts lagrangians* (tot i que tres d'ells són d'Euler), que es dentoten per $L_i$, $i = 1, \ldots, 5$. El punt $L_1$ (vegeu la figura 2) es pot entendre intuïtivament com el punt on les atraccions gravitatòries dels primaris es cancel·len. Si pensem que els primaris són el sol i la terra, el punt $L_1$ és un lloc ideal per enviar-hi un satèl·lit artificial que actuï com observatori solar. No obstant, aquest satèl·lit no pot estar situat exactament al punt $L_1$, perquè des de la terra es veuria dins del disc solar, i la radiació provinent del sol faria impossible la comunicació per ràdio. Per a aquestes missions es precisa d'òrbites en les quals, vistes des de la terra, el sol quedi enmig. Unes òrbites que compleixen aquesta propietat són unes òrbites periòdiques anomenades "òrbites halo". En l'actualitat, la missió SOHO (`http://sohowww.nascom.nasa.gov/`) es troba en una d'aquestes òrbites. Vegeu la figura 3.

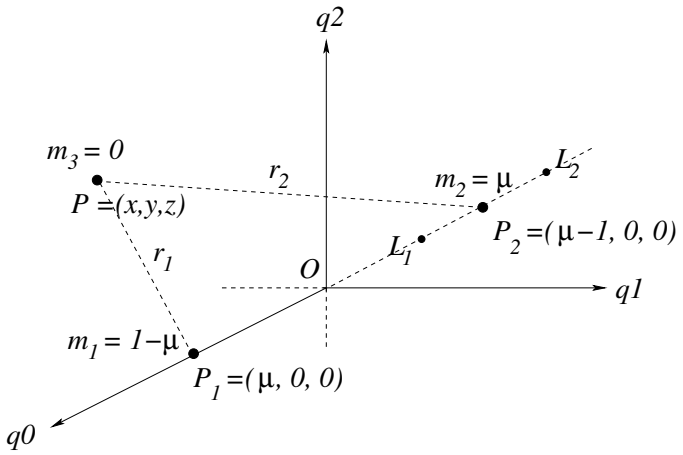


Figura 2: Primaris i punts de libració $L_1$ i $L_2$ del problema restringit de tres cossos.

El nostre problema consistirà a: donades 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 $\Delta t$, 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 $\Delta t/2$ unitats de temps, aplicar $\Delta \boldsymbol{v}_1$, i fer vol lliure durant $\Delta t/2$ unitats de temps més, acabem amb posició i velocitat $(\boldsymbol{r}_f, \boldsymbol{v}_f)^\top$.

# 1 Retrat-fase i flux d'un sistema d'EDO

Recordeu que, per a un sistema d'equacions diferencials ordinàries (EDO) de primer ordre i no autònom,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}), \tag{5}$$

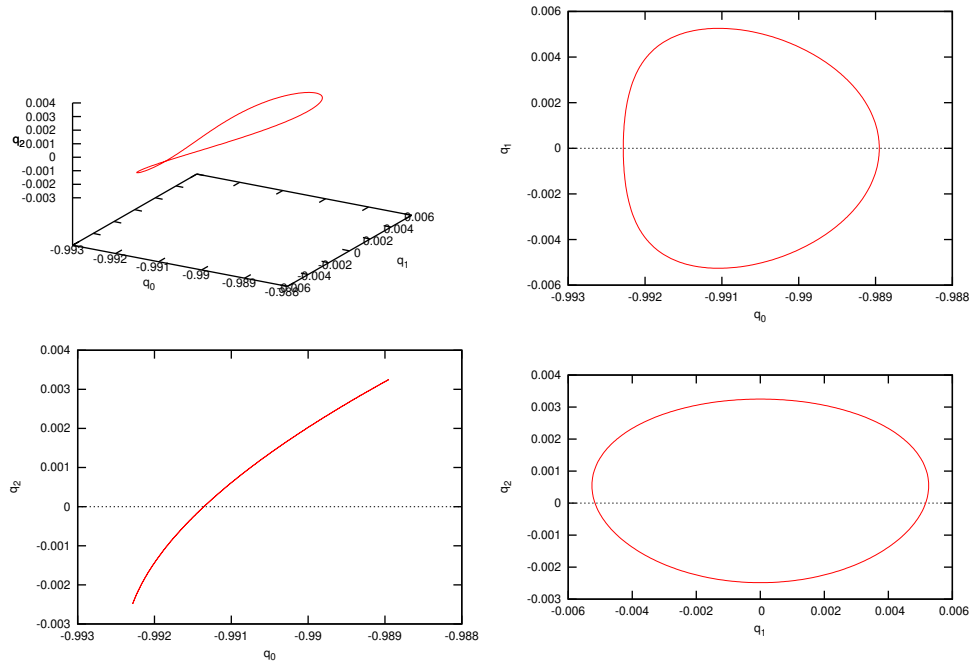


Figura 3: Projeccions $xyz$, $xy$ (vista des de dalt), $xz$ (vista de costat) i $yz$ (mirant de la terra al sol) d'una òrbita halo al voltant de $L_1$ del RTBP sol–terra. A la projecció $yz$, el disc solar quedaria dins l'òrbita, i d'aquí ve la denominació "òrbita halo".

on $\boldsymbol{x} \in \mathbb{R}^n$, $t \in \mathbb{R}$, $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$, sota hipòtesis de regularitat de $\boldsymbol{f}$ adequades, qualsevol problema de valors inicials per a aquesta equació té solució única. Es defineix llavors el flux de temps $t_0$ a temps $t$ com l'aplicació

$$\boldsymbol{x}_0 \longmapsto \boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0), \tag{6}$$

definida pel fet que l'aplicació $t \mapsto \boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0)$ és la solució del següent problema de valors inicials:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0) &= \boldsymbol{f}\big(t, \boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0)\big), \\ \boldsymbol{\phi}(t_0; t_0, \boldsymbol{x}_0) &= \boldsymbol{x}_0. \end{aligned}$$

Els mètodes de Runge–Kutta–Fehlberg (RKF) es troben entre els mètodes més emprats per integrar numèricament equacions diferencials ordinàries. Com heu vist a teoria, donats condicions inicials $(t_0, \boldsymbol{x}_0) \in \mathbb{R} \times \mathbb{R}^n$, un pas proposat $h_0$ i una tolerància $\mathrm{tol} > 0$, un mètode RKF torna $(t_1, \boldsymbol{x}_1) \in \mathbb{R} \times \mathbb{R}^n$ i un nou pas proposat $h_1$, satisfent:

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

Al campus virtual trobareu un fitxer `rk78.c` amb una funció `rk78()` que implementa un mètode RKF d'ordres 7 i 8. El seu prototipus és

```
int rk78 (double *t, double x[], double *h,
      double hmin, double hmax, double tol,
      int n, int (*camp)(int n, double t, double x[], double f[], void *prm),
      void *prm);
```

Dins `rk78.c` trobareu comentaris que expliquen el significat de cadascun dels arguments. Aquí només destacarem que l'argument `camp` és un punter a funció que implementa el camp vectorial $\boldsymbol{f}$ de (5).

## 1.1 Guió

**1.–** Feu un programa anomenat `rf_pendol.c` que, respongui a la crida

```
./rf_pendol hmin hmax tol
```

Aquest programa, per cada línia que rebi per standard input de la forma

```
x0 y0 h0 np
```

el que ha de fer és cridar `rk78() np` vegades, partint del punt (`x0`, `y0`), temps `t` = 0 i pas inicial `h0`. Abans de la primera crida i després de cada crida, ha d'escriure una línia de la forma

```
t x y h
```

on `t` és el temps actual, (`x`, `y`) la posició actual i `h` el pas proposat per la següent crida a `rk78()`. En acabar les `np` crides, ha d'escriure una línia en blanc.

**2.–** Empreu el problema anterior per tal de representar un retrat-fase del pèndol, similar al de la figura 1.

**3.–** Escriviu una funció C 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 $\boldsymbol{\phi}(t_0+T;t_0,\boldsymbol{x})$, 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 funció C ha de tornar $t_0+T$ a `*t`, $\boldsymbol{\phi}(t_0+T;t_0,\boldsymbol{x})$ a `x[]`, i a `*h` la proposta de pas per a crides posteriors. La resta de paràmetres són:

- `pasmin`, `pasmax`, `tol` són respectivament 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 ($\mu$ en el cas del RTBP, $\sigma, \rho, \beta$ en el cas del model de Lorenz més avall).

La funció `flux()` ha de tornar 0 si ha acabat amb èxit, $< 0$ si no (si `rk78()` ha tornat un error, o s'han superat `npasmx` passos d'integració numèrica).

**4.–** Verifiqueu la rutina anterior integrant el següent problema de valors inicials:

$$y' = 2y/t, \qquad y(1) = 1,$$

del qual la solució es $y(t) = t^2$.

Integreu també l'oscil·lador harmònic:

$$\begin{aligned} x' &= y, \\ y' &= -x. \end{aligned}$$

Assegureu-vos que els errors que obteniu són coherents amb la tolerància que demaneu.

**5.–** Considerem les *equacions de Lorenz*:

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= -xz + \rho x - y, \\ \dot{z} &= xy - \beta z. \end{aligned} \tag{7}$$

Aquest sistema d'EDO constitueix un model meteorològic simplificat que Lorenz va analitzar a un article del 1963. Té un atractor conegut com a *atractor de Lorenz* (veieu la figura 4), que constitueix un dels primers exemples de caos en el sentit de la teoria de sistemes dinàmics (i va donar lloc a l'expressió popular "efecte papallona").

Escriviu un programa `./lorenz_int`, que respongui a la crida

```
./lorenz_int sgm rho bet tf nt
```

que, per cada línia a standard input de la forma

```
x0 y0 z0
```

bolqui per standard ouptut

$$\begin{matrix} t_0 & \boldsymbol{\phi}(t_0; 0, \boldsymbol{x}^{(0)}) \\ t_1 & \boldsymbol{\phi}(t_1; 0, \boldsymbol{x}^{(0)}) \\ \vdots & \\ t_{n_t} & \boldsymbol{\phi}(t_{n_t}; 0, \boldsymbol{x}^{(0)}) \end{matrix}$$

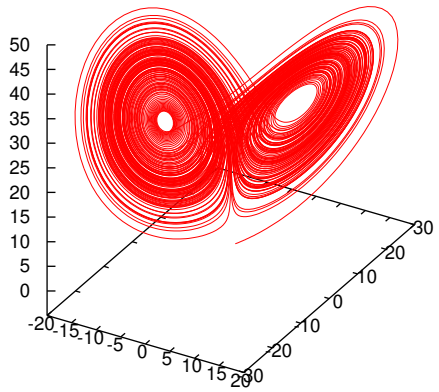


Figura 4: Atractor de Lorenz

on $\boldsymbol{x}^{(0)} = (x_0^{(0)}, x_1^{(0)}, x_2^{(0)})^\top \in \mathbb{R}^n$, $t_i = it_f/n_t$ i el flux $\boldsymbol{\phi}(t; t_0, \boldsymbol{x}^{(0)})$ correspon al sistema d'EDO (7). A les línies anteriors, cal afegir dos salts de línia (dues línies buides).

Representeu gràficament el atractor de Lorenz tal com apareix a la figura 4, per als paràmetres

$$\sigma = 10, \quad \rho = 28, \quad \beta = 2.6,$$

i partint del punt

```
x0=1.1 y0=0.36 z0=3.14
```

Suposant que la sortida del programa la teniu a un fitxer `lorenz.txt`, al qual les coordenades $x, y, z$ dels punts són a les columnes $2, 3, 4$, podeu representar l'atractor de Lorenz amb les ordres gnuplot:

```
set term x11 size 700,700
set view equal xyz
set ticslevel 0.1
unset key
splot 'lorenz.txt' u 2:3:4 w l
# ajusteu paràmetres i rotació perquè el gràfic quedi bé
set output 'lorenz.eps'
set term post eps color solid
replot
set term x11
set output
```

**6.–** Escriviu una utilitat similar per al RTBP, que respongui a la crida

```
./rtbps_int mu
```

on `mu` és el paràmetre $\mu$ a (4). 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`.

# 2 Les variacionals primeres i la diferencial del flux

Més endavant necessitarem avaluar la diferencial del flux respecte de condicions inicials,

$$D_{\boldsymbol{x}}\boldsymbol{\phi}(t;t_0,\boldsymbol{x}) = \Big(\partial_{x_0}\boldsymbol{\phi}(t;t_0,\boldsymbol{x}) \quad \partial_{x_1}\boldsymbol{\phi}(t;t_0,\boldsymbol{x}) \quad \dots \quad \partial_{x_{n-1}}\boldsymbol{\phi}(t;t_0,\boldsymbol{x})\Big),$$

on suposem $\boldsymbol{x} = (x_0,\dots,x_{n-1})^\top$. Per a obtenir-la, hem d'integrar el següent problema de valors inicials, que inclou les equacions variacionals primeres:

$$\left\{\begin{array}{rcl} \dot{\boldsymbol{x}} &=& \boldsymbol{f}(t,\boldsymbol{x}) \\ \dot{A} &=& D_{\boldsymbol{x}}\boldsymbol{f}(t,x)A \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0,\ A(t_0) &=& I_n\ , \end{array}\right. \tag{8}$$

on $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{f}:\mathbb{R}\times\mathbb{R}^n\to\mathbb{R}^n$, $A\in\mathcal{M}_{n\times n}(\mathbb{R})\approx\mathbb{R}^{n^2}$ i $I_n$ és la matriu identitat $n\times n$. Al PVI anterior, si denotem $\boldsymbol{f}(t,\boldsymbol{x}) = \big(f_0(t,\boldsymbol{x}),\dots,f_{n-1}(t,\boldsymbol{x})\big)^\top$, la part de les equacions diferencials s'expandeix com

$$\left\{\begin{array}{rcl} \dot{x}_0 &=& f_0(t,\boldsymbol{x}), \\ &\vdots& \\ \dot{x}_{n-1} &=& f_{n-1}(t,\boldsymbol{x}) \\ \begin{pmatrix} \dot{a}_{0,0} & \dots & \dot{a}_{0,n-1} \\ \vdots & & \vdots \\ \dot{a}_{n-1,0} & \dots & \dot{a}_{n-1,n-1} \end{pmatrix} &=& \begin{pmatrix} \frac{\partial f_0}{\partial x_0}(t,\boldsymbol{x}) & \dots & \frac{\partial f_0}{\partial x_{n-1}}(t,\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{\partial f_{n-1}}{\partial x_0}(t,\boldsymbol{x}) & \dots & \frac{\partial f_{n-1}}{\partial x_{n-1}}(t,\boldsymbol{x}) \end{pmatrix} \begin{pmatrix} a_{0,0} & \dots & a_{0,n-1} \\ \vdots & & \vdots \\ a_{n-1,0} & \dots & a_{n-1,n-1} \end{pmatrix} \end{array}\right. \tag{9}$$

Per a integrar numèricament aquestes equacions diferencials, s'han d'escriure en la forma

$$\dot{\boldsymbol{X}} = \boldsymbol{F}(t,\boldsymbol{X}).$$

Una manera de fer-ho és prenent

$$\boldsymbol{X} = (x_0, \ldots, x_{n-1}, a_{0,0}, \ldots, a_{n-1,0}, \ldots, a_{0,n-1}, \ldots, a_{n-1,n-1})^\top$$

(observeu que emmagatzemem $A$ **per columnes**) i $\boldsymbol{F}$ corresponent a aquesta ordenació. En components,

$$\begin{aligned} X_k &= x_k, & F_k(t, \boldsymbol{X}) &= f_k(t, \boldsymbol{x}), & \text{per a} \quad k = 0 \div n-1, \\ X_{n+jn+i} &= a_{i,j}, & F_{n+jn+i}(t, \boldsymbol{X}) &= \sum_{k=0}^{n-1} \frac{\partial f_i}{\partial x_k}(t, \boldsymbol{x}) a_{k,j}, & \text{per a} \quad i, j = 0 \div n-1. \end{aligned}$$

De cara a simplificar la implementació del camp $\boldsymbol{F}$ en una funció `camp()` per a passar-li a `rk78()`, observeu que

- Per a accedir a les components de $\boldsymbol{X}$ que corresponen als $a_{i,j}$, us podeu ajudar d'una macro amb arguments:

```
#define A(i,j) x[n+(j)*n+(i)]
```

- Podeu implementar els camps $\boldsymbol{f}$ i $\boldsymbol{F}$ simultàniament a la mateixa funció C amb l'ajuda de l'argument $n$: si és més petit que la dimensió espacial, només ompliu les primeres $n$ components de `f[]` (quart argument de `camp()`) amb les components de $\boldsymbol{f}(t, \boldsymbol{x})$. Si no, l'ompliu sencer d'acord amb (9).

## 2.1 Guió

1. Considereu el següent sistema d'EDO:

$$\begin{aligned} \dot{x} &= \alpha(1 - r^2)x - y, \\ \dot{y} &= x + \alpha(1 - r^2)y, \end{aligned} \tag{10}$$

   on $r^2 = x^2 + y^2$. Escriviu una funció C per tal d'integrar numèricament aquest sistema juntament amb les seves variacionals primeres (equació (9)).

   Verifiqueu numèricament que, llevat dels errors d'integració numèrica (fitats per la tolerància que passeu a `rk78()` a través de `flux()`), tenim

$$D\boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0) = \begin{pmatrix} X_2(t) & X_4(t) \\ X_3(t) & X_5(t) \end{pmatrix},$$

   on $(X_i(t))_{i=0}^5$ és la solució del problema de valors inicials (8), amb $\boldsymbol{f}$ corresponent al sistema (10). Només cal que ho feu per a una $t$ concreta. Una manera còmoda de fer-ho és fer servir la diferència finita centrada de primer ordre,

$$g'(a) = \frac{g(a + \delta) - g(a - \delta)}{2\delta} + O(\delta^2).$$

En el nostre cas, hem d'aplicar aquesta fórmula per a aproximar totes les derivades parcials,

$$D_{\boldsymbol{x}}\boldsymbol{\phi}(t;t_0,\boldsymbol{x}) = \Big(\frac{\partial\boldsymbol{\phi}(t;t_0,\boldsymbol{x})}{\partial x_j}\Big)_{j=0}^{n-1} = \left(\frac{\boldsymbol{\phi}(t;t_0,\boldsymbol{x}+\delta\boldsymbol{e}_j)-\boldsymbol{\phi}(t;t_0,\boldsymbol{x}-\delta\boldsymbol{e}_j)}{2\delta}\right)_{j=0}^{n-1} + O(\delta^2)\mathbf{1}_n,$$

on $\boldsymbol{e}_j$ és el $j$–èsim vector de la base canònica de $\mathbb{R}^n$ i $\mathbf{1}_n$ és la matriu $n\times n$ amb tots els coeficients iguals a 1.

Noteu que, per a avaluar les derivades numèriques, NO cal integrar les equacions variacionals.

Useu $\alpha = 0.4$, $t_0 = 0$ i $t$ no gaire grossa (per exemple 0.5).

# 3 Mètode QR per sistemes sobredeterminats

Volem trobar la solució per mínims quadrats de sistemes lineals sobredeterminats $Ax = b$, amb $A$ matriu $m\times n$, $m\geq n$, $b\in\mathbb{R}^m$ i $x\in\mathbb{R}^n$ desconegut. Recordeu que això vol dir trobar $x^*\in\mathbb{R}^n$ tal que

$$\|b - Ax^*\|_2 = \min_{x\in\mathbb{R}^n}\|b-Ax\|_2.$$

Ho farem aplicant transformacions de Householder a $A$ i $b$ fins què la matriu transformada de $A$ sigui triangular superior. Aleshores farem substitució enrere per a resoldre el sistema $n\times n$ format per les primeres $n$ equacions. Es pot veure amb un càlcul ràpid que això dóna la solució per mínims quadrats del sistema original.

Denotem

$$A^{(0)} = A, \qquad b^{(0)} = b.$$

Mitjançant l'aplicació de matrius de Householder, generarem una seqüència de matrius $A^{(1)}, A^{(2)},\ldots$ i vectors $b^{(1)}, b^{(2)},\ldots$ L'iterat $k$ d'aquest procés parteix d'una matriu $A^{(k)}$ i un vector $b^{(k)}$ de la forma

$$A^{(k)} = \left(\begin{array}{c|c} T^{(k)} & M^{(k)} \\ \hline 0 & \widetilde{A}^{(k)} \end{array}\right), \qquad b^{(k)} = \left(\begin{array}{c} c^{(k)} \\ \hline \widetilde{d}^{(k)} \end{array}\right) \tag{11}$$

on $T^{(k)}$ és un bloc $k\times k$ triangular superior, $M^{(k)}$ és un bloc $k\times(n-k)$, $\widetilde{A}^{(k)}$ és $(m-k)\times(n-k)$, $c^{(k)}\in\mathbb{R}^k$ i $\widetilde{d}^{(k)}\in\mathbb{R}^{m-k}$. Noteu que $A^{(0)} := A$ i $b^{(0)} := b$ tenen aquesta forma per a $k = 0$.

Denotem les components de $\widetilde{A}^{(k)}$ per

$$\widetilde{A}^{(k)} =: \begin{pmatrix} \widetilde{a}^{(k)}_{k,k} & \cdots & \widetilde{a}^{(k)}_{k,n-1} \\ \vdots & & \vdots \\ \widetilde{a}^{(k)}_{m-1,k} & \cdots & \widetilde{a}^{(k)}_{m-1,n-1} \end{pmatrix}.$$

Definim

$$\begin{aligned}
\gamma &= \left\| \left( \widetilde{a}^{(k)}_{k,k} \, , \, \widetilde{a}^{(k)}_{k+1,k} \, , \, \cdots \, , \, \widetilde{a}^{(k)}_{m-1,k} \right)^\top \right\|_2 \\
\widetilde{u}^{(k)} &= \left( \widetilde{a}^{(k)}_{k,k} + \operatorname{sig}(\widetilde{a}^{(k)}_{k,k})\gamma \, , \, \widetilde{a}^{(k)}_{k+1,k} \, , \, \cdots \, , \, \widetilde{a}^{(k)}_{m-1,k} \right)^\top \\
\widetilde{P}^{(k)} &= I_{m-k} - \frac{2}{\widetilde{u}^{(k)\top}\widetilde{u}^{(k)}} \widetilde{u}^{(k)}\widetilde{u}^{(k)\top}, \\
P^{(k)} &= \left( \begin{array}{c|c} I_k & 0 \\ \hline 0 & \widetilde{P}^{(k)} \end{array} \right)
\end{aligned}$$

on $I_j$ és la matriu identitat $j \times j$, $\widetilde{P}^{(k)}$ és $(m-k) \times (m-k)$ i $P^{(k)}$ és $m \times m$. Amb això calculem $A^{(k+1)}$ i $b^{(k+1)}$ com segueix:

$$A^{(k+1)} := P^{(k)}A^{(k)} = \left( \begin{array}{c|c} I_k & 0 \\ \hline 0 & \widetilde{P}^{(k)} \end{array} \right) \left( \begin{array}{c|c} T^{(k)} & M^{(k)} \\ \hline 0 & \widetilde{A}^{(k)} \end{array} \right) = \left( \begin{array}{c|c} T^{(k)} & M^{(k)} \\ \hline 0 & \widetilde{P}^{(k)}\widetilde{A}^{(k)} =: \overline{A}^{(k)} \end{array} \right)$$

i

$$b^{(k+1)} := P^{(k)}b^{(k)} = \left( \begin{array}{c|c} I_k & 0 \\ \hline 0 & \widetilde{P}^{(k)} \end{array} \right) \left( \begin{array}{c} c^{(k)} \\ \hline \widetilde{d}^{(k)} \end{array} \right) = \left( \begin{array}{c} c^{(k)} \\ \hline \widetilde{P}^{(k)}\widetilde{b}^{(k)} =: \overline{d}^{(k)} \end{array} \right).$$

Com que sabem que

$$\widetilde{P}^{(k)}\widetilde{a}^{(k)} = \begin{pmatrix} -\operatorname{sig}(\widetilde{a}^{(k)}_{k,k})\gamma \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

la matriu $\overline{A}^{(k)} = \widetilde{P}^{(k)}\widetilde{A}^{(k)}$ té zeros a la primera columna per sota de la primera fila, i per tant

$$\begin{aligned}
A^{(k+1)} &= \left( \begin{array}{c|c} T^{(k+1)} & M^{(k+1)} \\ \hline 0 & \widetilde{A}^{(k+1)} \end{array} \right), \\
b^{(k+1)} &= \left( \begin{array}{c} c^{(k+1)} \\ \hline \widetilde{d}^{(k+1)} \end{array} \right),
\end{aligned}$$

amb $T^{(k+1)}$ bloc $(k+1) \times (k+1)$ triangular superior, $M^{(k+1)}$ bloc $(k+1) \times (n-k-1)$, $\widetilde{A}^{(k+1)}$ bloc $(m-k-1) \times (n-k-1)$, $c^{(k+1)} \in \mathbb{R}^{k+1}$ i $\widetilde{d}^{(k+1)} \in \mathbb{R}^{m-k-1}$. Per tant, $A^{(k+1)}$ i $b^{(k+1)}$ tenen la mateixa estructura que $A^{(k)}$ i $b^{(k)}$ a (11).

En particular, $A^{(n)} = P^{(n-1)}P^{(n-2)} \dots P^{(0)}A^{(0)}$ tindrà la forma

$$A^{(n)} = \left( \begin{array}{c} T^{(n)} \\ \hline 0 \end{array} \right) =: R,$$

on $T^{(n)}$ és un bloc $n \times n$ triangular superior. Llavors

$$R = A^{(n)} = P^{(n-1)}P^{(n-2)} \dots P^{(0)}A \quad \Longrightarrow \quad A = P^{(0)}P^{(1)} \dots P^{(n-1)}R =: QR$$

(recordeu que les matrius $P^{(i)}$ són ortogonals simètriques), on $Q := P^{(0)}P^{(1)}\dots P^{(n-1)}$ és ortogonal i $R$ és triangular superior. A la nostra implementació, no arribarem a generar $Q$ explícitament, però sí que guardarem els vector $\widetilde{u}^{(0)},\dots,\widetilde{u}^{(n-1)}$ que defineixen les reflexions de Householder $P^{(0)},\dots,P^{(n-1)}$.

El procediment anterior es pot implementar en una rutina sobre una matriu de treball $m\times n$, que anomenarem `a[]`, i un vector $m$–dimensional, que anomenarem `b[]`. A cada pas, guardarem sobre la matriu de treball `a[]` la matriu $A^{(k)}$, i sobre el vector de treball `b[]` el vector $b^{(k)}$. Noteu que, a l'iterat $k$, tindrem zeros sota la diagonal a les $k$ primeres columnes de `a[]`. Podem aprofitar aquests zeros per a guardar els vectors $\widetilde{u}^{(0)},\dots,\widetilde{u}^{(k)}$ (recordeu que $\widetilde{u}^{(j)}\in\mathbb{R}^{m-j}$). Ens faltarà lloc per les primeres components de cada vector. Una manera còmoda de procedir és guardar-les a la diagonal de la matriu de treball. Això implica que haurem de guardar la diagonal de la matriu $A^{(k)}$ a un vector de treball addicional, que anomenarem `dr[]`.

Denotem

$$\widetilde{u}^{(k)} = \begin{pmatrix} \widetilde{u}^{(k)}_k \\ \vdots \\ \widetilde{u}^{(k)}_{m-1}\end{pmatrix},\quad c^{(k)} = \begin{pmatrix} c^{(k)}_0 \\ \vdots \\ c^{(k)}_{k-1}\end{pmatrix},\quad \widetilde{d}^{(k)} = \begin{pmatrix} \widetilde{d}^{(k)}_k \\ \vdots \\ \widetilde{d}^{(k)}_{m-1}\end{pmatrix},\quad \overline{d}^{(k)} = \begin{pmatrix} \overline{d}^{(k)}_k \\ \vdots \\ \overline{d}^{(k)}_{m-1}\end{pmatrix},$$

$$\widetilde{A}^{(k)} = \begin{pmatrix} \widetilde{a}_{k,k} & \cdots & \widetilde{a}_{k,n-1} \\ \vdots & & \vdots \\ \widetilde{a}_{m-1,k} & \cdots & \widetilde{a}_{m-1,n-1}\end{pmatrix},\qquad \overline{A}^{(k)} = \begin{pmatrix} \overline{a}_{k,k} & \cdots & \overline{a}_{k,n-1} \\ \vdots & & \vdots \\ \overline{a}_{m-1,k} & \cdots & \overline{a}_{m-1,n-1}\end{pmatrix}$$

$$M^{(k)} = \begin{pmatrix} m^{(k)}_{0,k} & \cdots & m^{(k)}_{0,n-1} \\ \vdots & & \vdots \\ m^{(k)}_{k-1,k} & \cdots & m^{(k)}_{k-1,n-1}\end{pmatrix}.$$

Amb aquesta notació podem detallar l'evolució de la matriu i vectors de treball durant la reducció de Householder descrita abans. A l'inici l'iterat $k$, el contingut de `a[]`, `b[]` i `dr[]` serà

$$\texttt{a[]} = \left(\begin{array}{cccc|cccc} \widetilde{\boldsymbol{u}}^{(\boldsymbol{0})}_{\boldsymbol{0}} & t^{(k)}_{0,1} & \cdots & t^{(k)}_{0,k-1} & m^{(k)}_{0,k} & m^{(k)}_{0,k+1} & \cdots & m^{(k)}_{0,n-1} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & t^{(k)}_{k-2,k-1} & \vdots & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}^{(\boldsymbol{0})}_{\boldsymbol{k-1}} & \cdots & \cdots & \widetilde{\boldsymbol{u}}^{(\boldsymbol{k-1})}_{\boldsymbol{k-1}} & m^{(k)}_{k-1,k} & m^{(k)}_{k-1,k+1} & \cdots & m^{(k)}_{k-1,n-1} \\ \widetilde{\boldsymbol{u}}^{(\boldsymbol{0})}_{\boldsymbol{k}} & \cdots & \cdots & \widetilde{\boldsymbol{u}}^{(\boldsymbol{k-1})}_{\boldsymbol{k}} & \widetilde{a}^{(k)}_{k,k} & \widetilde{a}^{(k)}_{k,k+1} & \cdots & \widetilde{a}^{(k)}_{k,n-1} \\ \widetilde{\boldsymbol{u}}^{(\boldsymbol{0})}_{\boldsymbol{k+1}} & \cdots & \cdots & \widetilde{\boldsymbol{u}}^{(\boldsymbol{k-1})}_{\boldsymbol{k+1}} & \widetilde{a}^{(k)}_{k+1,k} & \widetilde{a}^{(k)}_{k+1,k+1} & \cdots & \widetilde{a}^{(k)}_{k+1,n-1} \\ \vdots & & & \vdots & \vdots & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}^{(\boldsymbol{0})}_{\boldsymbol{m-1}} & \cdots & \cdots & \widetilde{\boldsymbol{u}}^{(\boldsymbol{k-1})}_{\boldsymbol{m-1}} & \widetilde{a}^{(k)}_{m-1,k} & \widetilde{a}^{(k)}_{m-1,k+1} & \cdots & \widetilde{a}^{(k)}_{m-1,n-1} \end{array}\right), \texttt{b[]} = \left(\begin{array}{c} c^{(k)}_1 \\ \vdots \\ c^{(k)}_{k-1} \\ \hline \widetilde{d}^{(k)}_k \\ \widetilde{d}^{(k)}_{k+1} \\ \vdots \\ \widetilde{d}^{(k)}_{m-1}\end{array}\right),$$

$$\texttt{dr[]} = \left(\begin{array}{cccc|ccc} t^{(k)}_{0,0} & \cdots & \cdots & t^{(k)}_{k-1,k-1} & * & * & \cdots & * \end{array}\right)$$

A la fi de l'iterat $k$, el seu contingut serà

$$\texttt{a[]} = \left(\begin{array}{cccc|ccc} \widetilde{\boldsymbol{u}}_{\boldsymbol{0}}^{(\boldsymbol{0})} & t_{0,1}^{(k)} & \cdots & t_{0,k-1}^{(k)} & m_{0,k}^{(k)} & m_{0,k+1}^{(k)} & \cdots & m_{0,n-1}^{(k)} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & t_{k-2,k-1}^{(k)} & \vdots & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k-1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k-1}}^{(\boldsymbol{k-1})} & m_{k-1,k}^{(k)} & m_{k-1,k+1}^{(k)} & \cdots & m_{k-1,n-1}^{(k)} \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{k})} & \overline{a}_{k,k+1}^{(k)} & \cdots & \overline{a}_{k,n-1}^{(k)} \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{k})} & \overline{a}_{k+1,k+1}^{(k)} & \cdots & \overline{a}_{k+1,n-1}^{(k)} \\ \vdots & & & \vdots & \vdots & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{k})} & \overline{a}_{m-1,k+1}^{(k)} & \cdots & \overline{a}_{m-1,n-1}^{(k)} \end{array}\right), \texttt{b[]} = \left(\begin{array}{c} c_1^{(k)} \\ \vdots \\ \vdots \\ c_{k-1}^{(k)} \\ \hline \overline{d}_k^{(k)} \\ \overline{d}_{k+1}^{(k)} \\ \vdots \\ \overline{d}_{m-1}^{(k)} \end{array}\right),$$

$$\texttt{dr[]} = \left(\begin{array}{cccc|ccc} t_{0,0}^{(k)} & \cdots & \cdots & t_{k-1,k-1}^{(k)} & t_{k,k}^{(k)} & * & \cdots & * \end{array}\right),$$

o, el que és el mateix,

$$\texttt{a[]} = \left(\begin{array}{cccccc|ccc} \widetilde{\boldsymbol{u}}_{\boldsymbol{0}}^{(\boldsymbol{0})} & t_{0,1}^{(k+1)} & \cdots & t_{0,k-1}^{(k+1)} & t_{0,k}^{(k+1)} & & m_{0,k+1}^{(k+1)} & \cdots & m_{0,n-1}^{(k+1)} \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots & & \vdots \\ \vdots & & \ddots & t_{k-2,k-1}^{(k+1)} & \vdots & & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k-1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k-1}}^{(\boldsymbol{k-1})} & t_{k-1,k}^{(k+1)} & & m_{k-1,k+1}^{(k+1)} & \cdots & m_{k-1,n-1}^{(k+1)} \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{k}}^{(\boldsymbol{k})} & & m_{k,k+1}^{(k+1)} & \cdots & m_{k,n-1}^{(k+1)} \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{k+1}}^{(\boldsymbol{k})} & & \widetilde{a}_{k+1,k+1}^{(k+1)} & \cdots & \widetilde{a}_{k+1,n-1}^{(k+1)} \\ \vdots & & & \vdots & \vdots & & \vdots & & \vdots \\ \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{0})} & \cdots & \cdots & \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{k-1})} & \widetilde{\boldsymbol{u}}_{\boldsymbol{m-1}}^{(\boldsymbol{k})} & & \widetilde{a}_{m-1,k+1}^{(k+1)} & \cdots & \widetilde{a}_{m-1,n-1}^{(k+1)} \end{array}\right), \texttt{b[]} = \left(\begin{array}{c} c_1^{(k+1)} \\ \vdots \\ \vdots \\ c_{k-1}^{(k+1)} \\ c_k^{(k+1)} \\ \hline \widetilde{d}_{k+1}^{(k+1)} \\ \vdots \\ \widetilde{d}_{m-1}^{(k+1)} \end{array}\right),$$

$$\texttt{dr[]} = \left(\begin{array}{cccccc|ccc} t_{0,0}^{(k+1)} & \cdots & \cdots & t_{k-1,k-1}^{(k+1)} & t_{k,k}^{(k+1)} & & * & \cdots & * \end{array}\right).$$

que serà el contingut de `a[]`, `b[]`, `dr[]` al començament de l'iterat $k+1$.

D'acord amb això, el procés de reducció de Householder es pot descriure mitjançant l'algorisme de la figura 5.

A l'hora d'implementar-lo, només cal tenir en compte a quines components de `a[]`, `b[]` i `tr[]` es refereixen cadascuna de les assignacions. Observeu que el bucle en $j$ d'aquest algorisme aplica $\widetilde{P}^{(k)}$ sobre $\widetilde{A}^{(k)}$ columna per columna. En efecte, si denotem les columnes de $\widetilde{A}^{(k)}$ per $\widetilde{a}_k^{(k)}, \ldots, \widetilde{a}_{n-1}^{(k)}$,

$$\begin{aligned} \widetilde{P}^{(k)}\widetilde{a}_j^{(k)} &= \Big(I_k - \frac{2}{\widetilde{u}^{(k)\top}\widetilde{u}^{(k)}}\widetilde{u}^{(k)}\widetilde{u}^{(k)\top}\Big)\widetilde{a}_j^{(k)} = \widetilde{a}_j^{(k)} - \frac{2}{\widetilde{u}^{(k)\top}\widetilde{u}^{(k)}}\widetilde{u}^{(k)}(\widetilde{u}^{(k)\top}\widetilde{a}_j^{(k)}) \\ &= \widetilde{a}_j^{(k)} - \frac{2}{\widetilde{u}^{(k)\top}\widetilde{u}^{(k)}}(\widetilde{u}^{(k)\top}\widetilde{a}_j^{(k)})\widetilde{u}^{(k)} = \widetilde{a}_j^{(k)} - \alpha\widetilde{u}^{(k)}, \end{aligned}$$

on, a la darrera igualtat, hem fet servir que $\beta = 2/(\widetilde{u}^{(k)\top}\widetilde{u}^{(k)})$ (comproveu aquesta igualtat!).

Si $m = n$, el sistema d'equacions lineals inicial $Ax = b$ és equivalent a $Rx = b^{(n)}$ ($b^{(n)} = P^{(n-1)}P^{(n-2)}\ldots P^{(0)}b$), i aquest sistema és triangular superior, de manera que el podem resoldre per substitució enrere. Per a $m \geq n$, es pot veure que la solució de $T^{(n)}x = c^{(n)}$ és la solució per mínims quadrats del sistema original $Ax = b$.

$$
\begin{array}{l}
\forall k = 0, 1, \ldots, n-1 \\
\qquad s := \mathrm{sig}(\widetilde{a}_{k,k}^{(k)}) \Big( \sum_{i=k}^{m-1} (\widetilde{a}_{i,k}^{(k)})^2 \Big)^{1/2} \\
\qquad \beta := 1/(s^2 + s\widetilde{a}_{k,k}^{(k)}) \\
\qquad \widetilde{u}^{(k)} := \left( \widetilde{a}_{k,k}^{(k)} + s, \widetilde{a}_{k+1,k}^{(k)}, \ldots, \widetilde{a}_{m-1,k}^{(k)} \right)^\top \\
\qquad t_{k,k}^{(k+1)} := -s \\
\qquad \forall j = k+1, k+2, \ldots, n-1 \\
\qquad\qquad \alpha := \beta \sum_{i=k}^{m-1} \widetilde{u}_i^{(k)} \widetilde{a}_{i,j}^{(k)} \\
\qquad\qquad \forall i = k, k+1, \ldots, m-1 \\
\qquad\qquad\qquad \overline{a}_{i,j}^{(k)} := \widetilde{a}_{i,j}^{(k)} - \alpha \widetilde{u}_i^{(k)} \\
\qquad \alpha := \beta \sum_{i=k}^{m-1} \widetilde{u}_i^{(k)} \widetilde{d}_i^{(k)} \\
\qquad \forall i = k, k+1, \ldots, m-1 \\
\qquad\qquad \overline{d}_i^{(k)} := \widetilde{d}_i^{(k)} - \alpha \widetilde{u}_i^{(k)}
\end{array}
$$

Figura 5: Reducció simultània d'una matriu $A$ i un terme independent $b$ mitjançant transformacions de Householder.

## 3.1 Guió

1. Implementeu l'algorisme anterior en una rutina amb el següent prototipus (observeu que la matriu `a[]` es guarda **per columnes**!). Els comentaris expliquen la seqüència d'arguments.

```
/*
 * Donada la matriu A m x n, fa la seva descomposició QR i,
 * opcionalment, resolt un sistema sobredeterminat associat.
 *
 * Arguments:
 * - m,n (e) : nombre de files i columnes de la matriu A.
 *   Restricció: m>=n.
 * - a (e/s) : En entrar, a[i+j*m], i=0..m-1, j=0..n-1 han de ser els
 *   coeficients de la matriu a (per files). En sortir:
 *   + Dins a[i+j*m], j>i, hi ha la part de sobre la diagonal de la R
 *     de la descomposició QR.
 *   + Dins a[i+k*m], i=k..m-1, hi ha les coordenades de la
 *     k-èsima reflexió de Householder que s'ha aplicat, k=0..n-1.
 * - dr (s) : en sortir conté la diagonal de R (dr[j] és el coeficient
 *   de la fila j, columna j de R, j=0..n-1).
 * - b (e/s) : En entrar: terme independent del sistema lineal
```

```
 *   sobredeterminat que es vol resoldre. En sortir:
 *         P(n-1) P(n-2) ... P(0) b
 *   on P(k) es la k-èsima reflexió de Householder.
 *   Si b==NULL, no es fa servir.
 * - x (s) : Solució del sistema sobredeterminat Ax=b.
 *   Si b==NULL, no es fa servir.
 */
void qrres (int m, int n, double *a, double *dr, double *b, double *x);
```

2. Verifiqueu la rutina anterior, amb el següent exemple

$$A = \begin{pmatrix} 0 & -4 \\ 0 & 0 \\ 5 & -2 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}.$$

   Les matrius i vectors intermedis i finals són

$$A^{(1)} = \begin{pmatrix} -5 & 2 \\ 0 & 0 \\ 0 & 4 \end{pmatrix}, \quad A^{(2)} = \begin{pmatrix} -5 & 2 \\ 0 & -4 \\ 0 & 0 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} -2 \\ 3 \\ -1 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} -2 \\ 1 \\ -3 \end{pmatrix},$$

$$\widetilde{u}^{(0)} = (5, 0, 5)^\top, \quad \widetilde{u}^{(1)} = (4, 4)^\top, \quad x = (0.3, -0.25)^\top.$$

3. Escriviu un programa principal que, donat $n$, generi una matriu aleatòria $n \times n$, triï $b$ perquè la solució sigui $(1, 1, \ldots, 1)^\top$ i resolgui el sistema $Ax = b$. Escriviu l'error màxim en la determinació de $x$,

$$\max_{i=0,\ldots,n-1} |x_i - 1|.$$

   Recordeu que es poden generar nombres aleatoris entre 0 i 1 fent

```
((double)rand())/RAND_MAX
```

   (la funció `rand()` està declarada a `stdlib.h`). Si voleu inicialitzar el generador de nombres aleatoris amb una llavor aleatòria (perquè no us doni sempre la mateixa seqüència), podeu fer, abans de generar el primer nombre aleatori (i només UNA vegada en tot el programa),

```
srand(time(0));
```

   (per a això necessiteu incloure el fitxer de capçalera `time.h`).

   Experimenteu amb diversos valors de $n$.

4. Retoqueu el programa principal de l'apartat anterior perquè escrigui el temps emprat en la descomposició $QR$ (crideu `qrres()` amb `b==NULL`, per tal que només faci la descomposició $QR$). Comproveu experimentalment que el nombre d'operacions és

$O(n^3)$. Tenint en compte que, més concretament, el nombre total d'operacions per fer la descomposició $QR$ és $4n^3/3 + O(n^2)$ (comptant suma i producte com operacions diferents), trobeu estimacions del rendiment efectiu del vostre programa. Compareu-les amb el pic teòric de la màquina.

Per a mesurar el temps d'execució del programa de manera portable, podeu usar la funció `clock()`, que està declarada a `time.h`. Aquesta funció torna el nombre de "ticks" que han transcorregut des de l'inici del programa. La macro `CLOCKS_PER_SEC` dóna el nombre de "ticks" per segon (que depèn de la implementació). Així, per a mesurar el temps transcorregut entre dos punts del programa, podeu fer

```
   time_t t0, t1;
/* Vull mesurar des d'aquí ... */
   t0=clock();
/* ... fins a aquí */
   t1=clock();
   printf("temps: %.3f s\n", ((double)(t1-t0))/CLOCKS_PER_SEC);
```

# 4 Càlcul de les maniobres

Reprenem la notació del flux (6). Donat que treballarem amb sistemes d'EDO de primer ordre autònomes ($\boldsymbol{f}(t, \boldsymbol{x})$ no depèn de $t$), denotarem

$$\boldsymbol{\phi}_t(\boldsymbol{x}_0) := \boldsymbol{\phi}(t; 0, \boldsymbol{x}_0).$$

Amb aquesta notació per al flux, el procés d'anar de $(\boldsymbol{r}_0, \boldsymbol{v}_0)^\top$ a $(\boldsymbol{r}_f, \boldsymbol{v}_f)^\top$ tot aplicant les maniobres $\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1$ el podem formular així:

$$\begin{aligned}
\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 \end{pmatrix} &\xrightarrow{\text{mani } \Delta\boldsymbol{v}_0} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} \\
&\xrightarrow{\text{vol lliure}} \boldsymbol{\phi}_{\Delta t/2} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} \\
&\xrightarrow{\text{mani } \Delta\boldsymbol{v}_1} \boldsymbol{\phi}_{\Delta t/2} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix} \\
&\xrightarrow{\text{vol lliure}} \boldsymbol{\phi}_{\Delta t/2} \left( \boldsymbol{\phi}_{\Delta t/2} \begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix} \right).
\end{aligned}$$

Com que volem que la darrera expressió sigui igual a $(\boldsymbol{r}_f, \boldsymbol{v}_f)^\top$, volem trobar maniobres

$$\Delta\boldsymbol{v} = \begin{pmatrix} \Delta\boldsymbol{v}_0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix} \in \mathbb{R}^n,$$

que anul·lin la funció

$$\boldsymbol{G}(\Delta\boldsymbol{v}) = \underbrace{\boldsymbol{\phi}_{\Delta t/2} \left( \underbrace{\boldsymbol{\phi}_{\Delta t/2} \underbrace{\begin{pmatrix} \boldsymbol{r}_0 \\ \boldsymbol{v}_0 + \Delta\boldsymbol{v}_0 \end{pmatrix}}_{=: \boldsymbol{x}_{0.5}} + \begin{pmatrix} 0 \\ \Delta\boldsymbol{v}_1 \end{pmatrix}} \right)}_{=: \boldsymbol{x}_{1.5}} - \begin{pmatrix} \boldsymbol{r}_f \\ \boldsymbol{v}_f \end{pmatrix}. \tag{12}$$

L'expressió $\boldsymbol{G}(\Delta\boldsymbol{v}) = \boldsymbol{0}$ és un sistema no lineal de $n = 2m$ equacions, que resoldrem pel mètode de Newton. Per a això caldrà avaluar la diferencial de $\boldsymbol{G}$, que és

$$D\boldsymbol{G}(\Delta\boldsymbol{v}) = \left( \; D_{\Delta\boldsymbol{v}_0}\boldsymbol{G}(\Delta\boldsymbol{v}) \;\middle|\; D_{\Delta\boldsymbol{v}_1}\boldsymbol{G}(\Delta v) \; \right). \tag{13}$$

Les expressions per als dos grups de $m$ columnes $D_{\Delta\boldsymbol{v}_0}\boldsymbol{G}(\Delta\boldsymbol{v})$, $D_{\Delta\boldsymbol{v}_1}\boldsymbol{G}(\Delta v)$ són

$$D_{\Delta\boldsymbol{v}_0}\boldsymbol{G}(\Delta\boldsymbol{v}) = D\boldsymbol{\phi}_{\Delta t/2}(\boldsymbol{x}_{1.5})D_{\boldsymbol{v}}\boldsymbol{\phi}_{\Delta t/2}(\boldsymbol{x}_{0.5}), \tag{14}$$
$$D_{\Delta\boldsymbol{v}_1}\boldsymbol{G}(\Delta\boldsymbol{v}) = D_{\boldsymbol{v}}\boldsymbol{\phi}_{\Delta t/2}(\boldsymbol{x}_{1.5}), \tag{15}$$

on $D_{\boldsymbol{v}}\boldsymbol{\phi}_{\Delta t/2}$ és la matriu formada per les $m$ darreres columnes de la diferencial del flux temps $\Delta t/2$.

## 4.1 Guió

1. 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 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 (12) com $D\boldsymbol{G}$ definida a les equacions (13),(14),(15). A la llista d'arguments:

   - `x0` (entrada) apunta a un vector de llargada `2*m` que conté $\boldsymbol{r}_0, \boldsymbol{v}_0$.
   - `xf` (entrada) apunta a un vector de llargada `2*m` que conté $\boldsymbol{r}_f, \boldsymbol{v}_f$.
   - `dt` (entrada) és $\Delta t$.
   - `dv` (entrada) apunta a un vector de llargada `2*m` que conté $\Delta\boldsymbol{v} = (\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1)$.
   - `g` (sortida) apunta a un vector de llargada `2*m` on s'ha de guardar $\boldsymbol{G}(\Delta\boldsymbol{v})$ (eq. (12)).
   - `dg` (sortida) apunta a un vector de llargada `(2*m)*(2*m)` on s'ha de guardar la matriu $D\boldsymbol{G}(\Delta\boldsymbol{v})$ (13) per columnes.
   - `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.

2. 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 $\Delta\boldsymbol{v} = (\Delta\boldsymbol{v}_0, \Delta\boldsymbol{v}_1)^\top$ que resolguin $\boldsymbol{G}(\Delta\boldsymbol{v}) = 0$, amb $\boldsymbol{G}$ definida a (12). 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 $\Delta\boldsymbol{v}$. A l'entrada són aproximacions inicials, i a la sortida han de ser les maniobres buscades, o sigui, solució de $\boldsymbol{G}(\Delta\boldsymbol{v}) = 0$ (eq. (12)).
- `tol,maxit` (entrada) són tolerància i màxim número d'iterats permesos per al mètode de Newton.

3. Comproveu les dues rutines anteriors resolent el següent problema per al pèndol: partint de $r_0 = 1$, $v_0 = 0$, volem trobar les maniobres $\Delta v_0$, $\Delta v_1$ 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 0.0998646 nc 0.0934007
cmani(): it 1 ng 0.00154523 nc 0.00184063
cmani(): it 2 ng 1.93848E-08 nc 2.51935E-08
cmani(): it 3 ng 6.79055E-16
dv[] -0.09269815705223261 0.006207868634240964
```

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 $2\pi$. 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.

4. 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 10=maxitnwt < cmani_rtbp.inp
```

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