CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica en C
Càlcul d'òrbites periòdiques de sistemes hamiltonians

Resum

L'objectiu d'aquesta pràctica és calcular numèricament òrbites periòdiques de sistemes hamiltonians. A aquest objectiu hi arribarem gradualment, tot desenvolupant les eines necessàries al llarg de les sessions de pràctiques.

Introducció

Un sistema d'equacions diferencials ordinàries de primer ordre

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1}$$

per a $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ i $n = 2m$, s'anomena **sistema Hamiltonià** si existeix una funció

$$\begin{array}{rcl} H : \mathbb{R}^m \times \mathbb{R}^m & \longrightarrow & \mathbb{R} \\ \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} & \longmapsto & H(\boldsymbol{q}, \boldsymbol{p}) \end{array}$$

tal que el sistema d'EDO és de la forma

$$\begin{array}{rcl} \dot{\boldsymbol{q}} & = & \dfrac{\partial H}{\partial \boldsymbol{p}}, \\ \dot{\boldsymbol{p}} & = & -\dfrac{\partial H}{\partial \boldsymbol{q}}. \end{array} \tag{2}$$

Al sistema anterior, les derivades parcials de H respecte dels vectors $\boldsymbol{q}$, $\boldsymbol{p}$ s'han d'entendre component a component: suposant $\boldsymbol{q} = (q_0, \ldots, q_{m-1})^\top$, $\boldsymbol{p} = (p_0, \ldots, p_{m-1})\top$,

$$\frac{\partial H}{\partial \boldsymbol{q}} = \begin{pmatrix} \partial H/\partial q_0 \\ \vdots \\ \partial H/\partial q_{m-1} \end{pmatrix}, \quad \frac{\partial H}{\partial \boldsymbol{p}} = \begin{pmatrix} \partial H/\partial p_0 \\ \vdots \\ \partial H/\partial p_{m-1} \end{pmatrix}.$$

Si definim

$$\boldsymbol{x} := \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix} \in \mathbb{R}^n, \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} \partial H/\partial \boldsymbol{q} \\ -\partial H/\partial \boldsymbol{p} \end{pmatrix} \in \mathbb{R}^n,$$

aleshores el sistema (2) es pot escriure en la forma (1).

Una **òrbita periòdica** del sistema d'EDO (1) és una trajectòria periòdica solució del sistema. Es pot definir a partir d'una **condició inicial** $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$. Denotem per

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

el flux del sistema d'EDO (1) (noteu que és autònom). Si, per a $T > 0$, $\boldsymbol{x}^{(0)}$ satisfà

$$\begin{aligned} \boldsymbol{\phi}_T(\boldsymbol{x}^{(0)}) &= \boldsymbol{x}^{(0)}, \\ \boldsymbol{\phi}_t(\boldsymbol{x}^{(0)}) &\neq \boldsymbol{x}^{(0)}, \qquad \forall t \in (0,T), \end{aligned} \tag{3}$$

aleshores $\boldsymbol{x}^{(0)}$ és una **condició inicial** d'una **òrbita periòdica** de **període** T.

Els sistemes Hamiltonians acostumen a estar associats a problemes físics en els que es descriu el moviment d'una partícula o sistema de partícules. L'espai de les coordenades $\boldsymbol{q}$, anomenades **posicions**, és l'espai on té lloc el moviment, i es coneix amb el nom d'**espai de configuracions**. Les coordenades $\boldsymbol{p}$ s'anomenen **moments**. L'espai total $\mathbb{R}^n$ de les coordenades $(\boldsymbol{q}, \boldsymbol{p})^\top$ es coneix com a **espai de fases**.

Un exemple de sistema Hamiltonià és el pèndol,

$$\begin{aligned} \dot{q} &= p, \\ \dot{p} &= -\sin q. \end{aligned} \tag{4}$$

Físicament, q és l'angle que forma el fil del pèndol amb la vertical, en radians. Aquest sistema es pot escriure en la forma (2) per a

$$H(q,p) \;=\; H_p(q,p) \;:=\; p^2/2 - \cos q.$$

A la figura 1 podeu trobar el seu retrat-fase (i.e., la representació gràfica d'unes quantes òrbites de les que se'n dedueix el comportament global del sistema).

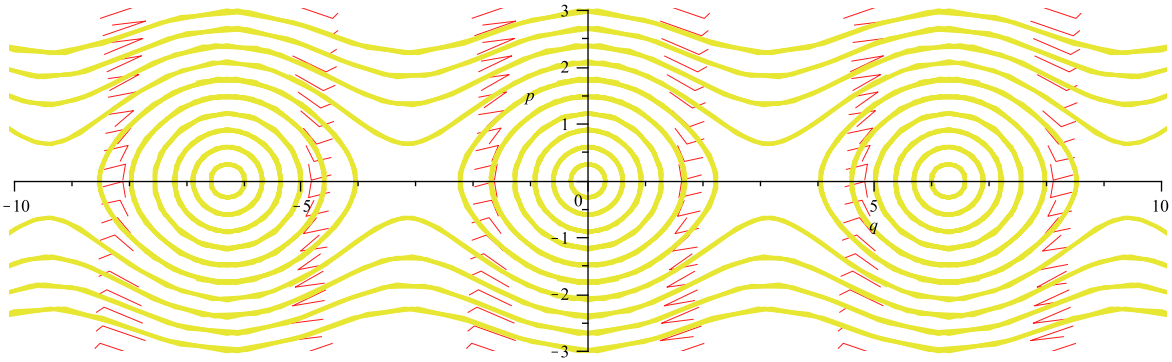

Figura 1: Retrat–fase del pèndol (eq. (4)).

L'objectiu d'aquesta pràctica és trobar òrbites periòdiques de sistemes Hamiltonians, com ara el pèndol. Observeu a la figura 1 que, al voltant de l'origen, hi ha tot d'òrbites periòdiques que arriben fins a la separatriu entre els moviments de libració i circulació (no representada). Els programes d'aquesta pràctica es poden fer servir per trobar aquestes òrbites.

De fet, no cal cap programa per trobar geomètricament les òrbites periòdiques del pèndol. Per a qualsevol $q_0 \in (0, \pi)$, el punt $\boldsymbol{x}^{(0)} = (q_0, 0)^\top$ és condició inicial d'una òrbita periòdica, que té energia $h_0 := H_p(q_0, 0) = -\cos q_0$. Com que H és l'energia i es manté constant al llarg de les trajectòries[1], de fet l'òrbita periòdica d'energia h_0 es pot descriure mitjançant l'equació

$$p^2/2 - \cos q = h_0.$$

El que no és trivial és trobar una expressió d'aquesta òrbita periòdica com a trajectòria, és a dir, q, p com a funció del temps. Els programes d'aquesta pràctica permeten trobar (numèricament) q, p com a funció del temps.[2]

Un altre model d'interès per nosaltres serà el *problema restringit de tres cossos circular espacial* (en anglès *spatial circular restricted three–body problem*, o RTBP). Aquest model descriu el moviment d'un cos de massa negligible sota l'atracció de dos cossos de masses $m_1 > m_2$ no negligibles, anomenats primaris. Els primaris se suposa que giren en cercles de manera uniforme un al voltant de l'altre. Quan es parla de missions espacials, el cos de massa negligible és una sonda espacial, i els cossos massius són o bé el sol i un planeta o bé un planeta i una lluna.

S'acostuma a prendre un sistema de coordenades giratori, en el qual els primaris estan fixats a l'eix x, l'origen és el centre de masses dels primaris, i el pla horitzontal correspon al pla de rotació dels primaris. Amb un canvi d'unitats adequat, els primaris passen a tenir masses $1-\mu$ i μ, per a $\mu = m_2/(m_1+m_2) \in [0, 1/2]$, passen a estar situats a $(\mu, 0, 0)$ i $(\mu - 1, 0, 0)$ (noteu que es troben a distància 1), i passen a completar una revolució en 2π unitats de temps. Les equacions del moviment del cos infinitesimal són, aleshores,

$$\begin{array}{rclcrcl} \dot{q}_0 &=& \partial H/\partial p_0, & \qquad & \dot{p}_0 &=& -\partial H/\partial q_0, \\ \dot{q}_1 &=& \partial H/\partial p_1, & & \dot{p}_1 &=& -\partial H/\partial q_1, \\ \dot{q}_2 &=& \partial H/\partial p_2, & & \dot{p}_2 &=& -\partial H/\partial q_2, \end{array}$$

amb

$$H(q_0, q_1, q_2, p_0, p_1, p_2) = \frac{1}{2}(p_0^2 + p_1^2 + p_2^2) - q_0 p_1 + q_1 p_0 - \frac{1-\mu}{r_1} - \frac{\mu}{r_2},$$

essent $r_1 = \sqrt{(q_0 - \mu)^2 + q_1^2 + q_2^2}$, $r_2 = \sqrt{(q_0 - \mu + 1)^2 + q_1^2 + q_2^2}$. El sistema d'EDO i el Hamiltonià anteriors estan implementats a les funcions `rtbp()` i `rtbp_h()` del fitxer `rtbp.c` que trobareu a la web de l'assignatura.

En aquesta formulació, el RTBP té 5 punts d'equilibri anomenats *punts lagrangians*[3], 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,

[1] Ho podeu comprovar ràpidament: deriveu $H(\boldsymbol{q}, \boldsymbol{p})$ respecte del temps, suposant que $\boldsymbol{q} = \boldsymbol{q}(t)$ i $\boldsymbol{p} = \boldsymbol{p}(t)$ són solució de (2). De fet, aquesta propietat (que H és constant al llarg de les solucions) és certa per qualsevol hamiltonià que **no depengui del temps**.

[2] De fet, per al pèdol es poden donar expressions per $q(t), p(t)$ que involucren integrals el·líptiques. No obstant, avaluar (numèricament) aquestes integrals el·líptiques és computacionalment més car que emprar els programes d'aquesta pràctica.

[3] Tot i que els punts L_1, L_2, L_3, anomenats colineals, els va descobrir Euler.

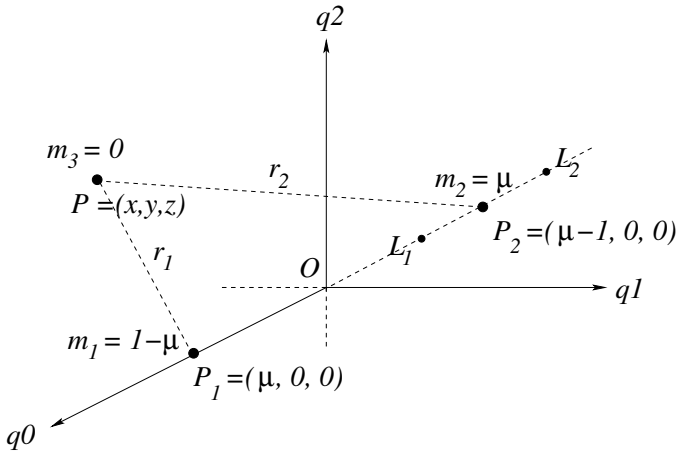

Figura 2: Primaris i punts de libració L_1 i L_2 del problema restringit de tres cossos.

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 ("òrbites halo"). En l'actualitat, la missió SOHO (`http://sohowww.nascom.nasa.gov/`) es troba en una d'aquestes òrbites. Vegeu la figura 3.

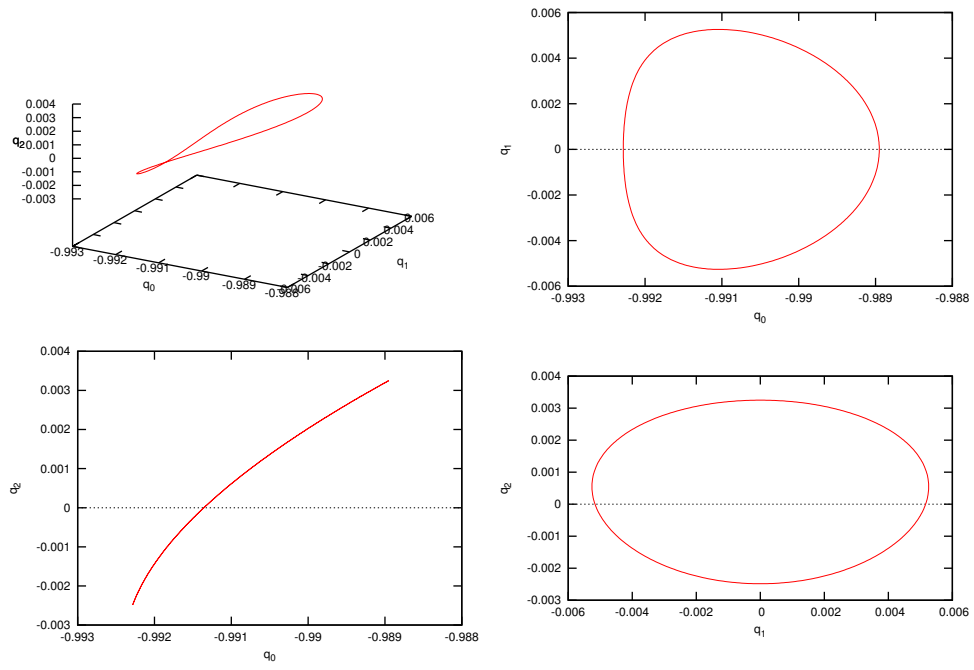

Figura 3: Projeccions $q_0q_1q_2$, q_0q_1 (vista des de dalt), q_0q_2 (vista de costat) i q_1q_2 (mirant de la terra al sol) d'una òrbita halo al voltant de L_1 del RTBP sol–terra. A la projecció q_0q_2, el disc solar quedaria dins l'òrbita, i d'aquí ve la denominació "òrbita halo".

CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica 1
Descomposició QR i resolució de sistemes sobredeterminats

Resum

L'objectiu d'aquesta pràctica és crear una rutina que ens permeti resoldre un sistema lineal $Ax = b$, posiblement sobredeterminat, fent servir la descomposició QR amb matrius de Householder.

Programari necessari: cap.
Programari resultant: `QRsolve`.

1 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 $A^{(0)} = A$ i fem n transformacions de la forma $Q_k A^{(k-1)} = A^{(k)}$ amb Q_k ortogonal, on $A^{(k)}$ té zeros a la columna k a sota la diagonal i les files i columnes $1, 2, \ldots, k-1$ de la matriu $A^{(k-1)}$ no s'han modificat al multiplicat per Q_k. Notem que això implica que la matriu $A^{(n)} = R$ és triangular superior. Llavors tenim:

$$R = A^{(n)} = Q_n A^{(n-1)} = \cdots = Q_n Q_{n-1} \cdots Q_1 A.$$

Definim $Q := Q_1^\top Q_2^\top \cdots Q_n^\top$. L'ortogonalitat de Q_k dóna:

$$QR = Q_1^\top Q_2^\top \cdots Q_n^\top Q_n \cdots Q_2 Q_1 A = A.$$

Les transformacions $Q_k A^{(k-1)} = A^{(k)}$ són:

$$\left(\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & P(\mathbf{u}^{(k)}) \end{array}\right) \left(\begin{array}{ccc|ccc} s_1 & a_{1,2}^{(1)} & \cdots & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ & s_2 & \cdots & a_{2,k}^{(2)} & \cdots & a_{2,n}^{(2)} \\ & & \ddots & \vdots & & \vdots \\ \hline & & & a_{k,k}^{(k-1)} & \cdots & a_{k,n}^{(k-1)} \\ & & & a_{k+1,k}^{(k-1)} & \cdots & a_{k+1,n}^{(k-1)} \\ & & & \vdots & & \vdots \\ & & & a_{m,k}^{(k-1)} & \cdots & a_{m,n}^{(k-1)} \end{array}\right) = \left(\begin{array}{ccc|ccc} s_1 & a_{1,2}^{(1)} & \cdots & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ & s_2 & \cdots & a_{2,k}^{(2)} & \cdots & a_{2,n}^{(2)} \\ & & \ddots & \vdots & & \vdots \\ \hline & & & s_k & \cdots & a_{k,n}^{(k)} \\ & & & 0 & \cdots & a_{k+1,n}^{(k)} \\ & & & \vdots & & \vdots \\ & & & 0 & \cdots & a_{m,n}^{(k)} \end{array}\right)$$

Denotem per $B^{(k-1)}$ i $B^{(k)}$ les submatrius d'abaix a la dreta de $A^{(k-1)}$ i $A^{(k)}$, respectivament. Definim:

$$\begin{aligned}
\mathbf{a}^{(k)} &= \left(a_{k,k}^{(k-1)} , a_{k+1,k}^{(k-1)} , \cdots , a_{m,k}^{(k-1)} \right)^\top \\
\mathbf{a}_j^{(k)} &= \left(a_{k,j}^{(k-1)} , a_{k+1,j}^{(k-1)} , \cdots , a_{m,j}^{(k-1)} \right)^\top , \ j = k+1, \ldots, n-1 \\
s_k &= -\text{signe}(a_{k,k}^{(k-1)}) \|\mathbf{a}^{(k)}\|_2 \\
\mathbf{u}^{(k)} &= \left(1, \frac{a_{k+1,k}^{(k-1)}}{a_{k,k}^{(k-1)} - s_k}, \cdots , \frac{a_{m,k}^{(k-1)}}{a_{k,k}^{(k-1)} - s_k} \right)^\top \\
\tau_k &= \frac{s_k - a_{k,k}^{(k)}}{s_k} \\
\mathbf{p}_j^{(k)} &= \mathbf{a}_j^{(k)} - \tau_k \left((\mathbf{u}^{(k)})^\top \mathbf{a}_j^{(k)} \right) \mathbf{u}^{(k)} \\
B^{(k)} &= (s_k \mathbf{e}_1, \mathbf{p}_{k+1}^{(k)}, \ldots, \mathbf{p}_{n-1}^{(k)}),
\end{aligned}$$

Substituint $B^{(k-1)}$ per $B^{(k)}$ tenim calculada $A^{(k)}$.

El procediment anterior es pot implementar en una rutina sobre una matriu de treball $m \times n$, que anomenarem `a[]`. A cada pas, guardarem sobre la matriu de treball `a[]` la matriu $A^{(k)}$. 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 $\mathbf{u}^{(0)}, \ldots, \mathbf{u}^{(k)}$. Observem que $\mathbf{u}^{(j)} \in \mathbb{R}^{m-j}$ i sota de la diagonal només hi ha $m-j-1$ espais. La utilitat de normalitzar els vectors $\mathbf{u}^{(j)}$ es veu aquí, ja que no cal guardar la primera component, que sempre és 1.

D'acord amb el procés explicat, la reducció de Householder es pot descriure mitjançant l'Algorisme 1 quan $m > n$. En el cas en que $m = n$, el loop inicial acaba un pas abans.

Fem el càlcul anàleg pel vector b. Definim $\mathbf{b}^{(0)} := b$ i $\mathbf{b}^{(k)} = Q_k \mathbf{b}^{(k-1)} = \left(\mathbf{b}_{1,\ldots,k-1}^{(k-1)} \mathbf{q}_{k\ldots m}^{(k)} \right)$ amb $\mathbf{q}_{k\ldots m}^{(k)} = \mathbf{b}_{k\ldots m}^{(k-1)} - \tau_k \left((\mathbf{u}^{(k)})^\top \mathbf{b}_{k\ldots m}^{(k-1)} \right) \mathbf{u}^{(k)}$. Llavors, $\mathbf{b}^{(n)} = Q^\top b$. Aquest procés està descrit a l'Algorisme 2.

Guió de la Pràctica 1

1. Implementeu els algorismes 1 i 2 en una rutina amb el prototipus
   ```
   int QRsolve (int m, int n, double *a, double *b, double *tau,...
                double *x, double tol);
   ```
   amb els arguments:

   - `m,n`: nombre de files i columnes de la matriu A. Restricció: $m \geq n$.
   - `a`: En entrar, `a[i+j*m], i=0..m-1, j=0..n-1` han de ser els coeficients de la matriu A (per columnes). En sortir: Dins `a[i+j*m], j>=i`, hi ha la matriu R de la descomposició QR. Dins `a[i+k*m], i=k+1..m-1`, hi ha les coordenades de la k-èsima reflexió de Householder que s'ha aplicat, $k = 0 \ldots n-1$.

Algorithm 1 Descomposició QR amb reemplaçament - Mètode de Householder

for $k = 1 \div n$ **do** ▷ Pas de la matriu $A^{(k-1)}$ a la matriu $A^{(k)}$
  $sk \leftarrow 0$;
  for $i = k \div m$ **do** ▷ Càlcul de $\|\mathbf{a}^{(k)}\|_2^2$
    $sk \leftarrow sk + a_{i,k}^2$;
  end for
  if $sk < tol$ **then**
    return -1; ▷ Detecció de matriu singular
  end if
  $sk \leftarrow \sqrt{sk}$; ▷ Càlcul de s_k
  if $a_{k,k} > 0$ **then**
    $sk \leftarrow -sk$;
  end if
  $\tau_k \leftarrow a_{k,k} - sk$; ▷ Usem τ_k com a variable auxiliar per emmagatzemar $a_{k,k} - s_k$
  for $i = k + 1 \div m$ **do**
    $a_{i,k} \leftarrow \frac{a_{i,k}}{\tau_k}$; ▷ Normalització del vector $\mathbf{u}^{(k)}$
  end for
  $\tau_k \leftarrow -\frac{\tau_k}{sk}$; ▷ Càlcul del multiplicador $\tau_k = \frac{s_k - a_{k,k}}{s_k}$
  $a_{k,k} \leftarrow sk$; ▷ Diagonal: hi emmagatzemem s_k
  for $j = k + 1 \div n$ **do** ▷ Arreglem la columna j
    $\beta \leftarrow a_{k,j}$; ▷ Inicialització de $\beta := (\mathbf{u}^{(k)})^\top \mathbf{a}_j^{(k)}$. Recordem que $(\mathbf{u}^{(k)})_1 = 1$
    for $i = k + 1 \div m$ **do**
      $\beta \leftarrow \beta + a_{i,k} a_{i,j}$; ▷ $\beta := (\mathbf{u}^{(k)})^\top \mathbf{a}_j^{(k)}$.
    end for
    $\beta \leftarrow \beta \tau_k$; ▷ $\beta := \tau_k (\mathbf{u}^{(k)})^\top \mathbf{a}_j^{(k)}$
    $a_{k,j} \leftarrow a_{k,j} - \beta$; ▷ Primera component de $P(\mathbf{u}^{(k)})\mathbf{a}_j^{(k)} = \mathbf{a}_j^{(k)} - \beta \mathbf{u}^{(k)}$; $(\mathbf{u}^{(k)})_1 = 1$
    for $i = k + 1 \div m$ **do**
      $a_{i,j} \leftarrow a_{i,j} - \beta a_{i,k}$; ▷ Les altres components de $P(\mathbf{u}^{(k)})\mathbf{a}_j^{(k)} = \mathbf{a}_j^{(k)} - \beta \mathbf{u}^{(k)}$
    end for
  end for
end for
return 0;

- `b`: En entrar: terme independent del sistema lineal sobredeterminat que es vol resoldre. En sortir: $Q^\top b$. Si `b==NULL`, no es fa servir.
- `tau`: en sortir conté τ_k a la posició k-èsima.
- `x`: en sortir conté la solució del sistema sobredeterminat $Ax = b$. Si `b==NULL`, no es fa servir.
- `tol`: tolerància per determinar la singularitat de la matriu A.

2. Escriviu un programa principal `Pr1Ex2.c` i verifiqueu la rutina anterior, amb el

Algorithm 2 Càlcul de $Q^\top b = Q_n \cdots Q_1 b$ a partir dels vectors $\mathbf{u}^{(k)}$

for $k = 1 \div n$ **do** ▷ Càlcul de $\mathbf{b}^{(k)} := Q_k \mathbf{b}^{(k-1)}$
  $\beta \leftarrow b_k$; ▷ Inicialització de $\beta := (\mathbf{u}^{(k)})^\top \mathbf{b}^{(k-1)}_{k\ldots m}$. Recordem que $(\mathbf{u}^{(k)})_1 = 1$
  for $i = k+1 \div m$ **do**
    $\beta \leftarrow \beta + a_{i,k} b_i$ ▷ $\beta := (\mathbf{u}^{(k)})^\top \mathbf{b}^{(k-1)}_{k\ldots m}$
  end for
  $\beta \leftarrow \beta \tau_k$; ▷ $\beta := \tau_k (\mathbf{u}^{(k)})^\top \mathbf{b}^{(k-1)}_{k\ldots m}$
  $b_k \leftarrow b_k - \beta$; ▷ Inici de $P(\mathbf{u}^{(k)})\mathbf{b}^{(k-1)}_{k\ldots m} = \mathbf{b}^{(k-1)}_{k\ldots m} - \beta \mathbf{u}^{(k)}$; $(\mathbf{u}^{(k)})_1 = 1$
  for $i = k+1 \div m$ **do**
    $b_i \leftarrow b_i - \beta a_{i,k}$ ▷ Les altres components de $P(\mathbf{u}^{(k)})\mathbf{b}^{(k-1)}_{k\ldots m} = \mathbf{b}^{(k-1)}_{k\ldots m} - \beta \mathbf{u}^{(k)}$
  end for
end for
return 0;

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 intermitjos i finals són

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

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

3. Escriviu un programa principal `Pr1Ex3.c` que resolgui el sistema lineal $Ax = b$ amb

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

4. Escriviu un programa principal `Pr1Ex4.c` 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.

CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica 2
Mètode de Newton

Resum

L'objectiu d'aquesta pràctica és crear una rutina que ens permeti resoldre una equació no lineal $F(x) = 0$ fent servir el mètode de Newton en diverses variables.

Programari necessari: `QRsolve`.
Programari resultant: `newton`.

2 Mètode de Newton

Suposem $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$ una aplicació diferenciable. Per resoldre l'equació

$$\mathbf{F}(\mathbf{x}) = 0$$

es pot plantejar un mètode iteratiu com és el de Newton:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (D\mathbf{F}(\mathbf{x}_n))^{-1}\mathbf{F}(\mathbf{x}_n),\ n \geq 0,$$

a partir d'una llavor inicial $\mathbf{x}_0$. En general, no es calcula $(D\mathbf{F}(\mathbf{x}_n))^{-1}$, sino que es resol el sistema lineal

$$D\mathbf{F}(\mathbf{x}_n)\mathbf{y}_n = \mathbf{F}(\mathbf{x}_n)$$

amb $\mathbf{y}_n = \mathbf{x}_n - \mathbf{x}_{n+1}$. Això és essencialment per tres motius:

- Donada una matriu A i un vector b, per tal de calcular $A^{-1}b$ és molt més eficient resoldre el sistema lineal $Ax = b$ que calcular la inversa de A i multiplicar per b.
- El càlcul de la inversa d'una matriu és un problema molt mal condicionat. En podeu veure un exemple a la pàgina 6 dels apunts
  `https://mat.uab.cat/~alseda/MatDoc/MetNumMat-LinAlg.pdf`
- Quan $n < m$, la matriu diferencial no és quadrada i per tant no és invertible. Això implica que el sistema lineal anterior és sobredeterminat.

Implementarem el mètode de Newton d'acord amb l'algorisme següent. Per resoldre el sistema lineal farem servir la rutina `QRsolve`.

Algorithm 3 Mètode de Newton

Entrada: h, $\mathbf{x}_0$, tol, maxit
for $k = 0 \div maxit$ **do**
  Avaluar $\mathbf{F}(\mathbf{x}_k)$ i $D\mathbf{F}(\mathbf{x}_k)$
  if $\|\mathbf{F}(\mathbf{x}_k)\|_2 < tol$ **then**
    STOP (èxit)
  end if
  Trobar $\mathbf{y}_k$ resolent el sistema lineal $D\mathbf{F}(\mathbf{x}_n)\mathbf{y}_n = \mathbf{F}(\mathbf{x}_n)$
  $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{y}_n$
end for
STOP (fracàs: no convergència)
Sortida: $\mathbf{x}_k$ (en cas d'èxit)

Guió de la Pràctica 2

1. Implementeu l'algorisme 3 en una rutina amb el prototipus

   ```
   int newton (int m, int n, double *x, int (*fdf)(int m, int n,...
       double *x, double *f, double *df),int maxit, double tol);
   ```

   amb els arguments:

   - `m`: nombre de components de la funció $\mathbf{F}$.
   - `n`: nombre de components de la solució $\mathbf{x}$.
   - `x`: En entrar, `x[i], i=0..n-1` han de ser les entrades de la llavor $\mathbf{x}_0$. En sortir, han de ser les entrades de la solució $\mathbf{x}_k$ proposada pel mètode de Newton.
   - `fdf`: Funció que calcula $\mathbf{F}$ i $D\mathbf{F}$.
   - `maxit`: Número màxim d'iterats que farà el mètode de Newton.
   - `tol`: Tolerància per decidir si el mètode de Newton ha trobat solució.

2. Escriviu un programa principal `Pr2Ex2.c` i verifiqueu la rutina anterior, amb el següent exemple

   $$\mathbf{F}(x_1, x_2, x_3) = (x_1 + x_2x_3, x_2 - x_1^2, \cos(x_3)).$$

   Podeu fer servir la funció `TestFunction.c` i la llibreria `TestFunction.h`. Com a funció `fdf` feu servir la funció `function1`.

3. El fractal de Newton és un conjunt frontera en el pla complex caracteritzat pel Mètode de Newton aplicat a un polinomi $p(z) \in \mathbb{C}[z]$. Quan no hi ha cicles atractors (d'ordre superior a 1), el fractal de Newton divideix el pla complex en regions G_k, cadascuna associada a una arrel ξ_k del polinomi $p(z)$. Un punt $z_0 \in G_k$ si, en fer el mètode de Newton

   $$z_{n+1} = z_n - \frac{p(z_n)}{p'(z_n)}$$

aquest dóna una sequència $\{z_k\}_{k\geq 0}$ que convergeix a ξ_k.

Considera $p(z) = z^3 - 1$. Anem a calcular el fractar de Newton de $p(z)$. Per tal d'aprofitar la feina feta, considerem $\mathbb{C} \approx \mathbb{R}^2$ amb $z = x + yi \to (x, y)$. D'aquesta manera, podem interpretar $p(z)$ com

$$p(x, y) = (x^3 - 3xy^2 - 1, 3x^2y - y^3).$$

Modifiqueu els arxius `TestFunction.c` i `TestFunction.h` per afegir l'aplicació anterior i la seva diferencial. Escriviu un programa principal `Pr2Ex3.c` que calculi, per a cada condició inicial $(x_0, y_0) \in [-1, 1] \times [-1, 1]$ en una malla (prou fina), en quina de les tres components G_k està (x_0, y_0). Guardeu les condicions inicials en tres fitxers, segons la component G_k on es trobi. Suposant que els arxius es denominen `g1.txt`, `g2.txt` i `g3.txt`, i que heu guardat les condicions inicials en files `x0 y0`, dibuixeu el fractal de Newton fent servir *gnuplot* amb la següent comanda:

```
plot "g1.txt" u 1:2 w dots lt rgb "red" notitle,...
     "g2.txt" u 1:2 w dots lt rgb "blue" notitle,...
     "g3.txt" u 1:2 w dots lt rgb "green" notitle
```

CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica 3
Mètode de Runge-Kutta

Resum

L'objectiu d'aquesta pràctica és crear una rutina que implementi un mètode de pas variable combinant dos mètodes de Runge-Kutta d'ordres 4 i 5, que ens permeti resoldre un problema de valor inicial.

Programari necessari: cap
Programari resultant: `rk45`

3 Mètode de pas fix

Donada $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ i els valors $t_0 \in \mathbb{R}$ i $\boldsymbol{x}_0 \in \mathbb{R}^n$, per a resoldre el problema de valor inicial

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}), \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \end{cases}$$

es pot plantejar el mètode de pas fix, h, per tal d'aproximar $\boldsymbol{x}(T)$.

$$\begin{array}{ll} t_0 & \boldsymbol{x}_0, \\ t_1 = t_0 + h, & \boldsymbol{x}_1 = \boldsymbol{x}_0 + h\boldsymbol{F}(t_0, \boldsymbol{x}_0, h; \boldsymbol{f}) \\ \vdots & \vdots \\ t_{k+1} = t_k + h, & \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h\boldsymbol{F}(t_k, \boldsymbol{x}_k, h; \boldsymbol{f}) \end{array}$$

Degut a que no sabem, a priori, els passos necessaris per arribar al valor T, és millor escollir un blucle tipus `while` per a finalitzar.

4 Mètodes de Runge-Kutta

Un pas dels mètodes de Runge-Kutta es pot escriure com

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h\boldsymbol{F}(t_k, \boldsymbol{x}_k, h; \boldsymbol{f}),\ k \geq 0$$

on la funció increment $\boldsymbol{F}$ es defineix com

$$\boldsymbol{F}(t_k, \boldsymbol{x}_k, h; \boldsymbol{f}) = \sum_{i=1}^{s} b_i \boldsymbol{K}_i,$$

amb

$$\boldsymbol{K}_i = \boldsymbol{f}(t_k + c_i h, \boldsymbol{x}_k + h \sum_{j=1}^{s} a_{ij} \boldsymbol{K}_j),\ i = 1, 2, \ldots, s$$

on s denota el número d'etapes del mètode. Els coeficients a_{ij}, c_i, b_i usualment s'escriuen de manera compacta usant la matriu de Butcher

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad \text{o} \quad \begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^\top \end{array}$$

5 Mètode de pas variable

Un mètode de pas variable, h_k, es pot plantejar a partir del mètode de pas fix anterior. Ara, donada $\boldsymbol{f}$, t_k, $\boldsymbol{x}_k$ i h_k,

$$t_{k+1} = t_k + h_k, \;\; \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h_k \boldsymbol{F}(t_k, \boldsymbol{x}_k, h_k; \boldsymbol{f})$$

El problema consisteix a trobar un bon mètode que permeti escollir el millor h_k a cada pas. En general, el pas s'escull dins un interval prefixat (h_{min}, h_{max}).

6 Pas adaptatiu pels Mètodes de Runge-Kutta d'ordres p i $p+1$

Un mètode de pas variable es pot fer combinant dos mètodes de Runge-Kutta d'ordres p i $p+1$. En aquest cas cal que, pels dos mètodes, les matrius A i els vectors $\boldsymbol{c}$ siguin iguals. Aquesta propietat es representa amb la matriu modificada de Butcher

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^\top \\ & \hat{\boldsymbol{b}}^\top \\ \hline & \boldsymbol{E}^\top \end{array}$$

D'aquesta manera s'aprofiten totes les avaluacions de les funcions pels dos mètodes. La matriu següent representa dos mètodes de Runge-Kutta d'ordres 4 i 5 de 6 etapes.

$$\begin{array}{c|cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\ \frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\ 1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\ \frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \\ \hline & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\ & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \\ \hline & \frac{1}{360} & 0 & -\frac{128}{4275} & -\frac{2197}{75240} & \frac{1}{50} & \frac{2}{55} \end{array}$$

Denotarem per $\boldsymbol{x}^p$ i $\boldsymbol{x}^{p+1}$ les aproximacions d'ordres p i $p+1$ calculades usant els coeficients $\boldsymbol{b}$ i $\hat{\boldsymbol{b}}$, respectivament.

Si denotem per $\mathcal{E}$ la tolerància absoluta que volem pel mètode, i fem servir com a solució exacta del problema la donada pel mètode d'ordre més alt, podem aproximar l'error al pas k per $e_k \approx \|\boldsymbol{x}_k^{p+1} - \boldsymbol{x}_k^p\|$. Així, prendrem com a nou valor del pas

$$h_{k+1} = \alpha h_k \sqrt[p+1]{\frac{\mathcal{E}}{e_k}},$$

on α s'acostuma a prendre 0.9. Si $|h_{k+1}| > h_{max}$ llavors es pren $h_{k+1} = \text{sgn}(h_{k+1}) h_{max}$.

En el cas en que $e_k \leq \mathcal{E}$, usem com a nou valor de $\boldsymbol{x}(t_{k+1})$ l'aproximació d'ordre $p+1$, $\boldsymbol{x}^{p+1}$. En cas contrari, repetim els càlculs fins que trobem un valor adequat per l'error. En el cas en que el mètode necessiti un pas inferior a h_{min}, es treu un missatge dient que no es pot calcular amb la precisió demanada.

Alternativament, es pot dissenyar un mètode en el que $\mathcal{E}$ denoti la tolerància relativa prenent

$$h_{k+1} = \alpha h_k \sqrt[p+1]{\frac{\mathcal{E} h_k}{e_k}}.$$

Guió de la Pràctica 3

1. Implementeu el mètode de Runge-Kutta de 6 etapes d'ordres 4 i 5 d'un pas adaptatiu donat per la taula anterior (fent servir $\hat{b}$), de manera que permeti resoldre qualsevol sistema d'equacions de dimensió n, amb el prototipus

   ```
   int rk45 (double *t, double *x, double *h, double hmin, double hmax,...
       double tol, int n, int (*camp)(int n, double t, double *x,...
       double *f, double *prm), double *prm);
   ```

   amb els arguments:

   - `t`: En entrar, temps t_k. En sortir, temps t_{k+1}.
   - `x`: En entrar, `x[i], i=0..n-1` han de ser les entrades de $\boldsymbol{x}_k$. En sortir, han de ser les entrades de $\boldsymbol{x}_{k+1}$.
   - `h`: En entrar, pas h_k. En sortir, pas h_{k+1}.
   - `hmin`: h_{min}.
   - `kmax`: h_{max}.
   - `tol`: tolerància absoluta $\mathcal{E}$.
   - `n`: dimensió del camp n.
   - `camp`: funció que calcula $\boldsymbol{f}(t, \boldsymbol{x})$ i guarda el valor al vector `f`.
   - `prm`: llista de paràmetres del camp $\boldsymbol{f}$ (si escau).

2. Escriviu un programa principal `Pr3Ex2.c` resolent el problema de valor inicial

$$\begin{cases} \dot{x} &= ax - y, & x(0) = x_0, \\ \dot{y} &= x + ay, & y(0) = y_0, \end{cases}$$

per $t \in [0, T]$ iterant la rutina `rk45` que respongui a la crida

```
Pr3Ex2 a T x0 y0
```

Compara la solució exacta d'aquest problema,

$$x(t) = e^{at}(x_0 \cos(t) - y_0 \sin(t)),\ y(t) = e^{at}(x_0 \sin(t) + y_0 \cos(t))$$

amb l'obtinguda amb `rk45` amb diferents rangs de pas pels casos següents:

- $t \in [0, 2\pi]$, $a = 0$.
- $t \in [0, 2\pi]$, $a = 0.1$.
- $t \in [0, -20\pi]$, $a = 0.1$.

Per fer la comparació, fes que el programa escrigui la solució exacta, la aproximada, i la diferència per a cada instant t i fes la gràfica fent servir `gnuplot`.

3. Escriviu un programa principal `Pr3Ex3.c` que trobi la solució aproximada del problema de valor inicials

$$\begin{cases} \dot{x} = \sigma(y - x), & x(0) = x_0, \\ \dot{y} = \rho x - y - xz, & y(0) = y_0, \\ \dot{z} = -\beta z + xy, & z(0) = z_0. \end{cases}$$

Per als valors $\beta = 8/3$, $\rho = 28$ i $\sigma = 10$ integra l'equació diferencial fent servir com a condició inicial un punt del primer octant per observar l'atractor de Lorenz amb `gnuplot`.

CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica 4
Flux d'una EDO i variacionals

Resum

L'objectiu d'aquesta pràctica és crear una rutina que calculi el flux d'una EDO i, alhora, calculi les solucions de les equacions variacionals primeres.

Programari necessari: `rk45`
Programari resultant: `flux`

7 Flux d'una EDO

Considerem un sistema d'equacions diferencials ordinàries de primer ordre i no autònom:

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

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, de manera que podem definir el flux de temps t_0 a temps t com l'aplicació

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

on l'aplicació $t \mapsto \boldsymbol{\phi}(t; t_0, \boldsymbol{x}_0)$ és la 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 i a la pràctica anterior, 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, un mètode RKF torna $(t_1, \boldsymbol{x}_1) \in \mathbb{R} \times \mathbb{R}^n$ i un nou pas proposat h_1, satisfent:

- $\|\boldsymbol{\phi}(t_1; t_0, \boldsymbol{x}_0) - \boldsymbol{x}_1\| < \text{tol}$,
- 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.

8 Variacionals primeres

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 (PVI), que inclou les equacions variacionals primeres:

$$\left\{\begin{array}{rcl} \dot{\boldsymbol{x}} &=& \boldsymbol{f}(t,\boldsymbol{x}) \\ \dot{A} &=& D_{\boldsymbol{x}}\boldsymbol{f}(t,\boldsymbol{x})A \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0,\ A(t_0) &=& I_n\ , \end{array}\right. \tag{5}$$

on $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{f}:\mathbb{R}\times\mathbb{R}^n\to\mathbb{R}^n$, $A\in\mathcal{M}_{n\times n}(\mathbb{R})$ 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

$$\left\{\begin{array}{rcl} \dot{x}_0 &=& f_0(t,\boldsymbol{x}), \\ &\vdots& \\ \dot{x}_{n-1} &=& f_{n-1}(t,\boldsymbol{x}) \\ \begin{pmatrix} \dot{a}_{0,0} & \dots & \dot{a}_{0,n-1} \\ \vdots & & \vdots \\ \dot{a}_{n-1,0} & \dots & \dot{a}_{n-1,n-1} \end{pmatrix} &=& \begin{pmatrix} \frac{\partial f_0}{\partial x_0}(t,\boldsymbol{x}) & \dots & \frac{\partial f_0}{\partial x_{n-1}}(t,\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{\partial f_{n-1}}{\partial x_0}(t,\boldsymbol{x}) & \dots & \frac{\partial f_{n-1}}{\partial x_{n-1}}(t,\boldsymbol{x}) \end{pmatrix} \begin{pmatrix} a_{0,0} & \dots & a_{0,n-1} \\ \vdots & & \vdots \\ a_{n-1,0} & \dots & a_{n-1,n-1} \end{pmatrix} \end{array}\right. \tag{6}$$

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

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

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

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

De cara a simplificar la implementació del camp $\boldsymbol{F}$ en una funció `camp()` per a passar-li a `rk45()`, 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 (6).

Guió de la Pràctica 4

1. Escriviu una funció amb prototipus

```
int flux (double *t, double *x, double *h, double T, double hmin,...
    double hmax, double tol, int npasmax, int n, ...
    int (*camp)(int n, double t, double *x, double *f, double *prm),...
    double *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:

   - `hmin`, `hmax`, `tol` són resp. el pas mínim, pas màxim i la tolerància per passar a `rk45()`,
   - El paràmetre `npasmax` és el nombre màxim de crides a `rk45()` 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.

   La funció `flux()` ha de tornar 0 si ha acabat amb èxit, < 0 si no (si `rk45()` ha tornat un error, o s'han superat `npasmax` passos d'integració numèrica).

   Tingueu en compte també que:

   - Heu d'admetre `T` negativa, i en aquest cas heu d'integrar temps enrere. Això vol dir que heu de vigilar que els signes de `T` i `*h` siguin coherents. Doneu prioritat a `T` (és a dir: canvieu el signe de `*h` si no és el de `T`).
   - El pas que deixeu a `*h` ha de ser adequat per poder tornar a cridar a `flux()` amb els valors finals de `*t`, `x[]` i `*h`, per tal d'avançar `T` unitats de temps més.

2. Verifiqueu la rutina anterior integrant les equacions de Lorenz de la pràctica anterior.

3. Considereu el següent sistema d'EDO:

$$\begin{aligned} \dot{x} &= \alpha(1-r^2)x - y, \\ \dot{y} &= x + \alpha(1-r^2)y, \end{aligned} \tag{7}$$

   on $r^2 = x^2+y^2$. Escriviu una funció `Pr4Ex3` per tal d'integrar numèricament aquest sistema juntament amb les seves variacionals primeres (equació (6)).

   Verifiqueu numèricament que, llevat dels errors d'integració numèrica (fitats per la tolerància que passeu a `rk45()` 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 (5), amb $\boldsymbol{f}$ corresponent al sistema (7). 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 $\alpha = 0.5$, $t_0 = 0$ i t no gaire grossa (per exemple 0.5).

CÀLCUL NUMÈRIC
Grau de Matemàtiques, 2024–2025

Pràctica 5
Càlcul d'òrbites periòdiques de sistemes hamiltonians

Resum

L'objectiu d'aquesta pràctica és trobar òrbites periòdiques de sistemes hamiltonians.

Programari necessari: `QRsolve`, `newton`, `rk45`, `flux`
Programari resultant: `opham_fdf`, `opham`

9 Càlcul d'òrbites periòdiques

Finalment, volem poder calcular numèricament òrbites periòdiques de sistemes Hamiltonians. Suposem donades unes equacions diferencials Hamiltonianes com a (2). Busquem $\boldsymbol{x}_0 \in \mathbb{R}^n$ condició inicial d'una òrbita periòdica de període T. Això vol dir que se satisfan les condicions de (3).

Ens podríem plantejar trobar una òrbita periòdica de període T buscant un zero de la funció

$$\boldsymbol{G}(\boldsymbol{x}) := \boldsymbol{\phi}_T(\boldsymbol{x}) - \boldsymbol{x}.$$

Això presenta dos inconvenients:

- Desconeixem quin període T fer servir (tot i que hi ha tècniques per fer-se una idea de quin és).
- El sistema d'equacions (no lineals) $\boldsymbol{G}(\boldsymbol{x}) = 0$ no té solucions aïllades. Si existeix una o.p. de període T, *tota* l'o.p. és solució de $\boldsymbol{G}(\boldsymbol{x}) = 0$. De fet, $\boldsymbol{G}(\boldsymbol{x}) = 0$ defineix implícitament l'o.p. com a varietat 1–dimensional. Donat $\boldsymbol{x}$ de l'o.p., el seu espai tangent és $\operatorname{Ker} D\boldsymbol{G}(\boldsymbol{x})$, que és un espai vectorial unidimensional. Això implica que $\det D\boldsymbol{G}(\boldsymbol{x}) = 0$, d'on no es pot calcular $D\boldsymbol{G}(\boldsymbol{x})^{-1}$, i tindríem un problema a l'hora d'implementar el mètode de Newton per trobar zeros de $\boldsymbol{G}$.

Aquests dos inconvenients es poden solucionar com segueix:

- En comptes de buscar òrbites d'un determinat període, podem buscar òrbites d'una determinada energia h. Per a això, només cal introduïr el període T com una incògnita addicional, i afegir l'equació $H(\boldsymbol{x}) - h = 0$.
- Per tal que les equacions tinguin un únic punt de l'òrbita com a solució, imposarem que la c.i. $\boldsymbol{x}$ que busquem estigui dins una hipersuperfície de la forma $\Sigma = \{g(\boldsymbol{x}) = 0\}$, per a $g : \mathbb{R}^n \to \mathbb{R}$. Per tal que això no doni problemes, cal demanar que la hipersuperfície sigui *transversal* al camp vectorial, això és, que $f(\boldsymbol{x}) \notin T_{\boldsymbol{x}}\Sigma$ per tot $\boldsymbol{x} \in \Sigma$, o, equivalentment, que $\nabla g(\boldsymbol{x}) \cdot f(\boldsymbol{x}) \neq 0$, on $\cdot$ denota el producte escalar euclidià. Una tal hipersuperfície s'anomena **secció de Poincaré** del camp $\boldsymbol{f}$.

Farem servir hiperplans. Per tant, considerarem funcions g de la forma

$$g(\boldsymbol{x}) = c_0 x_0 + c_1 x_1 + \cdots + c_{n-1} x_{n-1} + c_n, \tag{8}$$

on suposem

$$\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{p})^\top = (q_0, \ldots, q_{m-1}, p_0, \ldots, p_{m-1})^\top = (x_0, \ldots, x_{n-1})^\top.$$

Per exemple, per al pèndol, podem prendre

$$g(q, p) = p,$$

que correspon a triar $(c_0, c_1, c_2) = (0, 1, 0)$.

D'acord amb tot això, per trobar l'òrbita periòdica de nivell d'energia h del sistema d'EDO $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ amb Hamiltonià H, resoldrem el següent sistema d'equacions no lineals:

$$\left.\begin{array}{rcl} H(\boldsymbol{x}) & = & h \\ g(\boldsymbol{x}) & = & 0 \\ \boldsymbol{\phi}_T(\boldsymbol{x}) & = & \boldsymbol{x} \end{array}\right\}, \tag{9}$$

que denotarem de manera compacta com

$$\boldsymbol{F}(\boldsymbol{X}) = 0,$$

on $\boldsymbol{X} \in \mathbb{R}^{1+n}$ i $\boldsymbol{F} : \mathbb{R}^{1+n} \longrightarrow \mathbb{R}^{2+n}$ estan definits per

$$\boldsymbol{X} = \begin{pmatrix} T \\ \boldsymbol{x} \end{pmatrix} \in \mathbb{R}^{1+n}, \qquad \boldsymbol{F}(\boldsymbol{X}) = \begin{pmatrix} H(\boldsymbol{x}) - h \\ g(\boldsymbol{x}) \\ \boldsymbol{\phi}_T(\boldsymbol{x}) - \boldsymbol{x} \end{pmatrix} \in \mathbb{R}^{2+n}.$$

Per a la implementació de l'avaluació de $\boldsymbol{F}$, suposarem l'existència d'una rutina `ham()` que avalua el Hamiltonià H. L'avaluació de g la implementarem mitjançant la fórmula (8). Avaluarem el flux $\boldsymbol{\phi}_T(\boldsymbol{x})$ mitjançant la rutina `flux()` de la Pràctica 4.

De cara a implementar el mètode de Newton, necessitem la diferencial de $\boldsymbol{F}$, que és

$$D\boldsymbol{F}(x) = \left(\begin{array}{c|c} 0 & DH(\boldsymbol{x}) \\ 0 & Dg(\boldsymbol{x}) \\ \hline \boldsymbol{f}\big(\boldsymbol{\phi}_T(\boldsymbol{x})\big) & D\boldsymbol{\phi}_T(\boldsymbol{x}) - I \end{array}\right).$$

Per tal d'avaluar aquesta expressió, observem que:

- La rutina `camp()` permet avaluar $\boldsymbol{f}(\boldsymbol{x})$, i també $DH(\boldsymbol{x})$, donat que a partir de (2) es veu que

  $$DH(\boldsymbol{x}) = \big(-f_m(\boldsymbol{x}), -f_{m+1}(\boldsymbol{x}), \ldots, -f_{n-1}(\boldsymbol{x}), f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x})\big),$$

  suposant que $\boldsymbol{f}(x) = \big(f_0(\boldsymbol{x}), \ldots, f_{n-1}(\boldsymbol{x})\big)^\top$.

- $Dg(\boldsymbol{x})$ ve donada pels coeficients de l'hiperplà (8):

$$Dg(\boldsymbol{x}) = (c_0, \ldots, c_{n-1}).$$

- $D\boldsymbol{\phi}_T(\boldsymbol{x})$ s'obté integrant les equacions variacionals:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}), \\ \dot{A} &= D\boldsymbol{f}(\boldsymbol{x})A, \end{aligned}$$

amb condicions inicials $\boldsymbol{x}(0) = \boldsymbol{x}^{(0)}$, $A(0) = I$, on $I \in \mathcal{M}_{n\times n}$ és la matriu identitat. Aleshores,

$$D\boldsymbol{\phi}_T(\boldsymbol{x}) = A(T).$$

Per tal de resoldre $\boldsymbol{F}(\boldsymbol{X}) = 0$, feu servir la rutina `newton` del mètode de Newton de la Pràctica 2.

Guió de la Pràctica 5

1.– Programeu una rutina amb prototipus

```
int opham_fdf (
   int m, double hh, double *xx, double *cg, double *f, double *df,
   double h0rk, double hminrk, double hmaxrk, double tolrk, int npasmxrk,
   int (*ham)(double *x, double *h, double *prm),
   int (*camp)(int n, double t, double *x, double *f, double *prm),
   double *prm);
```

on `m` $= m$, `hh` $= h$ (nivell d'energia), `xx` $= \boldsymbol{X} \in \mathbb{R}^{1+n}$ (període i condició incial), `*cg` conté els coeficients $(c_0, \ldots, c_{n-1}, c_n)$ de l'hiperplà (8), `ham` és un punter a funció que implementa el Hamiltonià H (l'argument `*x` de `ham()` conté les coordenades del punt $\boldsymbol{x}$ on es vol trobar el valor del Hamiltonià, que s'ha de tornar dins `*h`), i `camp` és un punter a una funció que implementa el camp $\boldsymbol{f}$ i les seves variacionals primeres, tal com ho fèiem a la Pràctica 4.

La funció `opham_fdf()` ha de tornar dins `*f` el valor $\boldsymbol{F}(\boldsymbol{X})$, i, si `dff!=NULL`, ha de tornar dins `*dff` la diferencial $D\boldsymbol{F}(\boldsymbol{X})$, guardada **per columnes** (tal com l'espera `QRsolve()`). Si `dff==NULL`, `opham_fdf()` no ha de fer servir `dff`.

Els paràmetres `hminrk`, `hmaxrk`, `tolrk`, `npasmxrk`, són per passar a `flux()`. El paràmetre `h0rk` s'ha de prendre com a pas inicial per a la integració numèrica.

Disposeu dels fitxers `opham.c` i `opham.h` a la web de l'assignatura.

2.– Per tal de verificar la rutina anterior, verifiqueu mitjançant derivació numèrica que la matriu tornada dins `*dff` és la diferencial de la funció tornada dins `*f`, de manera anàloga a com ho feiem a la Pràctica 4.

Aquesta comprovació la podeu fer per al pèndol (4), per a $x^{(0)} = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$ i $T = 0.1$.

Disposeu dels fitxers `pendol.h` i `pendol.c` a la web de l'assignatura (heu d'agafar els de la part 4).

3.– Programeu una rutina amb prototipus

```
int opham (
   int m, double hh, double *xx, double tol, int maxit, double *cg,
   double h0rk, double hminrk, double hmaxrk, double tolrk, int npasmxrk,
   int (*ham)(double *x, double *h, double *prm),
   int (*camp)(int n, double t, double *x, double *f, double *prm),
   double *prm);
```

on `m` $= m$, `hh` $= h$ (nivell d'energia), `xx` $= \boldsymbol{X}^{(0)} \in \mathbb{R}^{1+n}$ (període i condició inicial aproximats), `*cg` conté els coeficients de l'hiperplà $(c_0, \ldots, c_{n-1}, c_n)$, `ham` és un punter a funció que implementa el Hamiltonià H (l'argument `*x` de `ham()` conté les coordenades del punt $\boldsymbol{x}$ on es vol avaluar el Hamiltonià), i `camp` és un punter a una funció que implementa el camp $\boldsymbol{f}$ i les seves variacionals primeres, de manera anàloga a la primera pràctica. Els arguments `tol` i `maxit` són els mateixos que a `newton()`.

La funció `opham` ha de trobar, a partir de $\boldsymbol{X}^{(0)} \in \mathbb{R}^{1+n}$, el període i una condició incial dins l'hiperplà definit pels coeficients $c_0, \ldots, c_n$ de l'òrbita periòdica d'energia h del sistema Hamiltonià definit per `ham()` i `camp()`. Per a això, ha de fer servir el programa `newton()`. En cas d'èxit, ha de tornar $\boldsymbol{X}^{(k)}$ dins `*xx` i el nombre d'iterats emprats via `return`. En cas de no convergència, ha de tornar el darrer iterat dins `*xx` i -1 via `return`.

Feu que, a cada iterat de Newton, escrigui el nombre d'iterat, $\|\boldsymbol{F}(\boldsymbol{X}^{(k)})\|_2$ i $\|\Delta\boldsymbol{X}^{(k)}\|_2$. Per exemple:

```
opham(): it 0 nf 0.00040621 nc 1.08646E-06
opham(): it 1 nf 1.67035E-07 nc 1.08125E-06
opham(): it 2 nf 1.04319E-09 nc 2.36645E-11
opham(): it 3 nf 1.33993E-12
```

4.– Comproveu la rutina anterior buscant l'òrbita periòdica del pèndol (4) d'energia $h = H_p(0.5, 0) = -\cos(0.5)$. Preneu com a aproximació inicial del període $T = 2\pi$. Un cop la tingueu, proveu de pertorbar tant el període com la condició inicial, i comproveu que la convergència és quadràtica.

Podeu anomenar `opham_main_pendol` el programa principal d'aquest punt, i fer que rebi una línia de comandes semblant a la del programa `opham_main_rtbp` del punt següent.

5.– Trobeu el període i una condició inicial sobre l'hiperplà $\Sigma = \{q_1 = 0\}$ de l'òrbita halo del RTBP terra-sol ($\mu = 3.040357143 \cdot 10^{-6}$) d'energia -1.500384. Empreu $T^{(0)} = 3.051858$ com a aproximació inicial del període, i

$$\boldsymbol{x}^{(0)} = (-0.988950, 0, 0.003235, 0, -0.999225, 0)$$

com a condició inicial aproximada. Demaneu una tolerància de 10^{-10} per al mètode de Newton. Empreu una tolerància de 10^{-13} per la integració numèrica.

Ho podeu fer mitjançant un programa `opham_main_rtbp.c`, que respongui a la crida

```
./opham_main_rtbp mu tol maxit cg[0] cg[1] cg[2] cg[3] cg[4] cg[5] cg[6]
```

on `mu` $= \mu$, `tol,maxit` siguin las del mètode de Newton, i (`cg[0]`, $\ldots$, `cg[6]`) $= (c_0, \ldots, c_6)$ són els coeficients de l'hiperplà (8). Li podeu passar h, T, $\boldsymbol{x}^{(0)}$ per standard input.

Heu d'obtenir:

$$\begin{aligned} T &= 3.05177804298575, \\ \boldsymbol{x} &= (-0.9889514941464619, 0, 0.003249420704787417, 0, -0.9992369544112847, 0). \end{aligned}$$

Representeu gràficament l'òrbita resultant (i.e., reproduïu la figura 3). Ho podeu fer amb `gnuplot`, mitjançant les ordres

```
splot 'orb.txt' u 2:3:4 w l
plot 'orb.txt' u 2:3 w l
plot 'orb.txt' u 2:4 w l
plot 'orb.txt' u 3:4 w l
```

suposant que heu guardat dins un fitxer `orb.txt` els punts de l'òrbita. Per a això, podeu escriure un programa `rtbp_int.c`, que faci servir la funció `flux()` per calcular l'òrbita amb les condicions inicials que heu obtingut i el període.