Introducció

La principal intenció d'aquestes pràctiques és continuar la vostra formació de Càlcul Numèric a través de l'anàlisi d'alguns problemes, la presentació de diferents tècniques i la seva implementació en un llenguatge de programació.

Abans d'entrar en matèria volem fer-vos memòria de dues eines bàsiques que necessitarem al llarg del curs com són la lectura/escriptura de fitxers de dades i la seva representació gràfica. La primera d'elles us la presentem a través d'un exemple senzill programat amb `C` (concretament `gcc`, compilador Linux de la casa Gnu)[1].

```
#include "math.h"
#include"stdio.h"
#define PI 3.14159265358979311599796346 9
FILE *fitxer;

int main(void) {
  int i;
  double x,fx;
  fitxer=fopen("prova","w");
  // Obrim el fitxer de dades en mode escriptura "w" (recordeu
  // que teniu altres modes:
  // r ..... de lectura
  // a..... afegeix a un fitxer ja creat
  // a+..... crea o afegeix per lectura i/o escriptura,
  // per mode texte, entre d'altres, aixi com analegs per mode binari.

  // comprovem que la reserva de memoria ha estat correcte
  if (fitxer==NULL) {
    printf("\n Error en obrir el fitxer de dades.\n");
    exit(1);
  }

  for(i=1;i<=20;i++) {
    x=PI/i;
    fx=sin(x);
    // escrivim al fitxer
    fprintf(fitxer,"%g   %g  \n",x,fx);
  }
```

[0] T. Lázaro, M. Ollé i J.R. Pacha. Departament Matemàtica Aplicada I

[1] Vosaltres podeu fer servir el seu equivalent `C++`

```
  fclose(fitxer);

  // Un cop tancat el fitxer, el tornem a obrir i llegirem les dades
  // que hem escrit anteriorment.

  fitxer=fopen("prova","r");

  // Comprovem de nou que la reserva de memoria ha estat correcte
  if (fitxer==NULL) {
    printf("\n Error en obrir el fitxer de dades.\n");
    exit(2);
  }

  // Lectura de dades i les escrivim per pantalla
  for(i=1;i<=20;i++) {
    fscanf(fitxer,"%g   %g  \n",&x,&fx);
    printf("x=%g  \t fx=%g  \n",x,fx);
    }

  // Tanquem el fitxer
  fclose(fitxer);

  return 0;
}
```

En quant a l'apartat gràfic, farem servir el programa `gnuplot`, d'ús lliure, amb versions Windows i Linux, i força estès. Us podeu descarregar el programa de (gnu). Un bon tutorial (reduït i per a la versió 4.2 però suficient) es troba, per exemple, a la següent pàgina a la Universitat de Duke: (tutorial). Tot i que es pot utilitzar de manera interactiva és corrent crear un programa que l'executi quan es vol fer repetidament. Aquest tipus de programes s'acostumen a anomenar *shells* (i n'hi ha, per Linux i Unix, de diversos tipus: `sh, bash, csh, tcsh, dash, ...`). Per exemple, crearem una *shell* que dibuixi les gràfiques del sinus i del cosinus superposades. Per fer-ho suposarem que tenim un fitxer anomenat `sinus-cosinus.dat` amb tres columnes: x, $\sin x$ i $\cos x$, per a valors d'x equiespaiats entre 0 i 2π (per exemple, $x_k = kh$ amb $h = 2\pi/1000$). El nostre fitxer *shell* es dirà `dibuix.gp`[2]:

```
#!/bin/sh
gnuplot << END
```

[2]L'extensió *gp* ens permet recordar que són instruccions `gnuplot` però també hauríem pogut posar *.sh*

```
# Determinem la sortida del grafic. En aquest cas la sortida sera
# a un fitxer postscript anomenat "grafiques.ps"
set terminal postscript
set output "grafiques.ps"

# Si haguessim volgut que la sortida fos directament per pantalla escriuriem
# set terminal x11

# Dibuixem les grafiques del sinus i del cosinus a [-1,1]x[-pi,pi]

plot [-3.142:3.142][-1:1] "sinus-cosinus.dat" u 1:2 title "sin(x)" w p pt 7 ps 0.6,
  "sinus-cosinus.dat" u 1:3 t "cos(x)" w p lt 3 pt 7 ps 0.6

# u 1:2  es l'abreviatura de 'using  1:2
# Amb aixo estem dient que dibuixi la segona columna (sin(x))
# del fitxer respecte de la primera (x)


# pt: point type  /  lt: line type
# ps: point size  /  ls: line size
# t: title  / notitle: sense titol
# w p: with points / w l: with lines

END
```

Per executar aquest programa feu

```
perico@delospalotes:  sh dibuix.gp
```

El fitxer `grafiques.ps` el podeu obrir amb qualsevol visualitzador de *postscript* (`okular, gv, ghostview`, etc).

Exercici -1.1 L'aplicació estàndard *o* Standard map *es la següent aplicació al pla:*

$$\begin{cases} y_{n+1} &= y_n + K\sin(x_n), \\ x_{n+1} &= x_n + y_{n+1} \end{cases} \qquad \text{mod } (2\pi) \tag{1}$$

Aleshores, considereu la finestra $[0, 2\pi] \times [0, 2\pi]$ *i preneu valors inicials de la forma* $(x_0, 0)$ *i* $(0, y_0)$ *a on* $x_0 = y_0 = 2j\pi/30$*,* $j = 0, 1, \ldots, 30$*.*

Fixeu, per exemple, el valor $K = 1.11$*. Per a cada punt inicial* $(x_0, 0)$*,* $(0, y_0)$ *calculeu* 300 *dels seus iterats per (1), o sigui, una part de les seves òrbites, i deseu-los en un fitxer* `estandard.dat` *de dues columnes* (x, y)*. Representeu-les usant* `gnuplot`*. Repetiu el càlcul anterior per altres valors de* $K \in (0.8, 3.4)$*.*

1 Mètodes iteratius per a sistemes lineals. Precondicionadors

Els mètodes iteratius que tractarem en aquesta pràctica provenen de l'utilització de l'anomenat *precondicionador*, una matriu procedent de la descomposició de la matriu del sistema, A, que proporciona a l'esquema recurrent final una estructura de càlcul triangular. Com a recordatori, si volem resoldre el sistema $Ax = b$, $A \in \mathcal{M}_{n,n}$, podem descomposar $A = P + (A - P)$ i escriure

$$Ax = b \Leftrightarrow Px = (P - A)x + b \Leftrightarrow x = (\mathrm{I} - P^{-1}A)x + P^{-1}b,$$

que deriva en l'esquema recurrent:

$$x^{(k+1)} = Bx^k + \tilde{b}, \tag{1}$$

on $B = (\mathrm{I} - P^{-1}A)$ és la matriu d'iteració i $\tilde{b} = P^{-1}b$. Si escrivim $A = L + D + U$, amb D la part diagonal de la matriu, L la part triangular estricta inferior i U la triangular estricta superior, aleshores els casos $P = D$ i $P = L + D$ corresponen als coneguts mètodes de Jacobi i Gauss-Seidel, respectivament.

L'esquema (1) és convergent si i només si el radi spectral de la matriu B, $\rho(B) = \max_{\lambda \text{ vap}} |\lambda|$, és estrictament menor que 1. Del fet que $\rho(B) \leq \|B\|$ per a qualsevol norma matricial es dedueix que $\|B\| < 1$ és una condició suficient per a la convergència del mètode iteratiu (1).

Exercici 1.1 *Considereu una matriu $A = (a_{ij})_{ij} \in \mathcal{M}_{m,n}$. Aleshores, escriviu programes - que puguin ser cridats des de qualsevol altre - que calculin:*

$$\|A\|_1 = \max_{j=1\ldots n} \sum_{i=1}^{m} |a_{ij}|, \qquad \|A\|_\infty = \max_{i=1\ldots m} \sum_{j=1}^{n} |a_{ij}|, \qquad \|A\|_2 = \left(\rho(A^\top A)\right)^{1/2}.$$

Per calcular la norma sub-2 d'una matriu necessitareu programar el càlcul del radi espectral $\rho(A)$ d'una matriu quadrada $A \in \mathcal{M}_{n,n}$. Per fer-ho implementeu el *mètode de la potència* (vegeu, per exemple, [1, 2, 3, 4]), que proporciona el vap de mòdul màxim. La idea d'aquest mètode és la següent: ordenem els vaps de la matriu A segons la seva multiplicitat i mòdul

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \cdots |\lambda_n|.$$

Suposem, per simplicitat, que A diagonalitza en una certa base de veps $u_1, u_2, \ldots, u_n$ i que existeix un únic vap de mòdul màxim: $|\lambda_1| > |\lambda_2| \geq \cdots$. Si prenem un vector inicial

$$x^{(0)} = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n, \qquad \alpha_1 \neq 0,$$

aleshores, en aplicar-li A obtindrem

$$Ax^{(0)} = \lambda_1\alpha_1 u_1 + \lambda_2\alpha_2 u_2 + \cdots + \lambda_n\alpha_n u_n.$$

[0] T. Lázaro, M. Ollé i J.R. Pacha. Departament Matemàtica Aplicada I

Si ho fem de manera iterada i anomenem $x^{(k)} = Ax^{(k-1)}$, aleshores

$$x^{(k)} = \lambda_1^k \alpha_1 u_1 + \lambda_2^k \alpha_2 u_2 + \cdots + \lambda_n^k \alpha_n u_n = \lambda_1^k \left(\alpha_1 u_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k \alpha_2 u_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k \alpha_n u_n \right)$$

Observeu que si fem k gran, el vector $x^{(k)}$ tendeix cap a la direcció del vep de valor propi λ_1. Si denotem $x^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right)$ llavors, sempre que tinguin sentit els quocients, tindrem

$$\begin{aligned} \frac{x_i^{(k+1)}}{x_i^{(k)}} &= \lambda_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right) \\ \frac{x^{(k)}}{\lambda_1^k} &= \alpha_1 u_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right). \end{aligned}$$

Noteu que la velocitat de convergència cap a λ_1 depén fortament del quocient $|\lambda_1/\lambda_2|$, de forma que si aquests dos primers vaps són similars aquesta pot ser molt lenta.

La successió $(\lambda_1^k)_k$ generalment tendeix cap a 0 o esdevé no afitada. Per evitar els problemes numèrics que això implica convé normalitzar a cada pas el vector $x^{(k)}$. Així, un possible algorisme per fer-ho seria: un cop tenim $x^{(k)}$, el normalitzem

$$y^{(k)} = \frac{x^{(k)}}{\|x^{(k)}\|}$$

i calculem el següent iterat $x^{(k+1)} = Ay^{(k)}$. Aleshores,

$$\lim_{k \to +\infty} \frac{x_i^{(k+1)}}{y_i^{(k)}} = \lambda_1, \qquad i = 1, \ldots, n$$

$$\lim_{k \to +\infty} y^{(k)} = \pm \frac{u_1}{\|u_1\|}.$$

Exercici 1.2 (Opcional) *Com ordena un cercador com Google les pàgines web que fan referència a una pregunta que hàgiu introduït ? Quin criteri segueix?*

La resposta està basada en un problema d'Àlgebra lineal, matrius de Markov i, òbviament, mètodes numèrics eficients que permetin calcular el valor propi dominant i un seu vector propi de matrius de mida realment gegant (de l'ordre de 10^9).

Si us interessa el problema i altres relacionats i us tempta programar-ho (a una escala més petita, és clar) descarregueu-vos l'article "El secreto de Google y el Álgebra Lineal" de Pablo Fernández (www.uam.es/personal_pdi/ciencias/gallardo/fernandez1.pdf), de la Universidad Autónoma de Madrid. Aquest article es publicà al Boletín de la Sociedad Española de Matemática Aplicada - SEMA - (volum 30 (2004), 115-141) i va guanyar el "V Premio SEMA a la Divulgación en Matemática Aplicada" l'any 2004.

Exercici 1.3 *Programeu la resolució numèrica d'un sistema $Ax = b$, $A \in \mathcal{M}_{n,n}$, fent servir els mètodes de Jacobi i de Gauss-Seidel. A cada programa caldria que féssiu:*

1. *Lectura de la matriu A d'un fitxer.*
2. *Cada un d'aquests programes hauria de rebre la matriu A, el seu nombre n de fileres (o columnes, doncs és quadrada) i els vectors x i b.*
3. *Calculeu les normes $\|\cdot\|_1$ i del suprem de la matriu d'iteració B corresponent i indiqueu, en el cas de que alguna d'elles sigui menor estricta que 1, la convergència del mètode. En el cas que ambdues siguin més grans o iguals a 1, calculeu el radi espectral de B, $\rho(B)$.*
4. *Si el mètode convergeix, obteniu un valor aproximat $\tilde{x}$ de la solució amb una precisió determinada* `TOL` *(per exemple, 10^{-12}) i en un nombre màxim d'iteracions* `NUM_MAX_ITER` *que també fixareu a priori. Un cop tingueu calculada aquesta aproximació haurieu d'escriure per pantalla informació sobre una estimació de l'error en l'aproximació $\|x^{(K)} - x^{(K-1)}\|_\infty$, on $x^{(K)} = \tilde{x}$.*
5. *Calculeu la norma del suprem del residu: $\|A\tilde{x} - b\|_\infty$.*

Exercici 1.4 (Extra) *Fer el mateix amb un mètode de sobre-relaxació (*SOR*).*

Exercici 1.5 *Resoleu numèricament (usant Jacobi i Gauss-Seidel) els següents sistemes $Ax = b$. Compareu la velocitat de convergència en ambdós casos.*

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

$$A = \begin{pmatrix} 0.95 & 0.07 & 0. & 0. & 0.05 & 0.01 \\ 0.07 & 0.95 & 0.07 & 0. & 0. & 0.04 \\ 0. & 0.07 & 0.95 & 0.06 & 0. & 0. \\ 0. & 0. & 0.06 & 0.95 & 0.06 & 0. \\ 0.05 & 0. & 0. & 0.06 & 0.95 & 0.06 \\ 0.01 & 0.04 & 0. & 0. & 0.06 & 0.95 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

A molts problemes d'elements finits o diferències finites és necessària la resolució de (grans) sistemes lineals $Ax = b$ on la matriu A és una matriu per blocs. Per exemple, si $T \in \mathcal{M}_{m,m}$ és la matriu tridiagonal simètrica:

$$T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}$$

i I la matriu identitat m-dimensional, aleshores considerem

$$A = \begin{pmatrix} T & -I & & \\ -I & T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & T \end{pmatrix}, \tag{2}$$

matriu tridiagonal simètrica per blocs. Emprar un mètode de resolució directa a A (per exemple, Choleski) seria poc eficient ja que A és *escasa* i moltes de les operacions que faríem involucrarien coeficients nuls.

El que es fa sovint és aprofitar aquesta estructura per blocs i transformar el sistema $Ax = b$ en un grapat de sistemes més petits, molts d'ells amb la mateixa matriu (en el nostre cas serien del tipus $T\bar{x} = \bar{b}$ doncs $-I\bar{x} = \bar{b}$ és de resolució immediata).

Exercici 1.6 *Apliqueu els mètodes de Jacobi i de Gauss-Seidel per a la resolució d'un sistema $Ax = b$, amb A del tipus* (2) *amb $N \times N$ blocs. Feu-ho sense descomposar-lo en subsistemes petits, és a dir, treballant amb tota la matriu A. (Nota: no feu ara la lectura dels coeficients de la matriu A des d'un fitxer sino definiu-los vosaltres directament en funció dels valors d'm i N, que podeu llegir per pantalla.)*

Bibliografia

[1] A. Aubanell, A. Benseny, A. Delshams. *Eines bàsiques de càlcul numèric*. Universitat Autònoma de Barcelona, Barcelona, 1991.

[2] C. Bonet, À. Jorba, T.M-Seara, J. Masdemont, M. Ollé, A. Susin, M. Valencia. *Càlcul numèric*. Aula Teòrica 23, Edicions UPC, Barcelona, 1994.

[3] G. Dahlquist, A. Björck. *Numerical methods*. Prentice Hall Inc., New Jersey, 1974.

[4] J. Stoer, R. Bulirsch. *Introduction to numerical analysis* (third edition). Texts in Applied Mathematics, Springer-Verlag, New York - Berlin - Heidelberg, 2002.

2 Sistemes sobredeterminats. Aproximació per mínims quadrats

S'anomena *sistema sobredeterminat* a un sistema lineal en el qual el nombre d'observacions m és major que el nombre n d'incògnites, és a dir, del tipus

$$Ax = b, \qquad A \in \mathcal{M}_{m\times n}, \quad m > n.$$

Aquests sistemes, típicament, no tenen solució. El que es busca en canvi es trobar aquella solució x tal que minimitzi $\|Ax - b\|_2$. Per aquest motiu es coneix com a *aproximació per mínims quadrats*. Una manera usual de resoldre'l és projectant el vector b sobre l'espai vectorial generat pels vectors columna de la matriu A:

$$A^\top Ax = A^\top b. \tag{1}$$

L'expressió (1) s'anomena *Equacions Normals*. Si la matriu inicial A té rang màxim (o sigui n), aleshores la matriu $A^\top A \in \mathcal{M}_{n\times n}$ és regular. Malgrat això, és fàcil trobar exemples de matrius A de rang màxim que en aritmètica de coma flotant de doble precisió proporcionen $A^\top A$ molt propera a singular.

Fins i tot en el cas de que $A^\top A$ sigui regular aplicar un mètode gaussià per a resoldre (1) no és tampoc el més convenient en general. Concretament, atès que $A^\top A$ és simètrica, seria raonable aplicar una descomposició de Choleski $A^\top A = LL^\top$, amb L matriu triangular inferior. La dificultat numèrica prové del fet que el *nombre de condició* de la matriu $A^\top A$ i el de la matriu L poden créixer considerablement.

Un manera de preservar un mínim control sobre el nombre de condició $\mu_2(A) = \|A\|_2\|A^{-1}\|_2$ es basa en l'anomenada *descomposició* QR de la matriu A: escrivim $A = QR$, on Q és ortogonal (unitària si és complexa) i R triangular superior. Així,

$$A^\top Ax = A^\top b \Leftrightarrow (QR)^\top (QR)x = (QR)^\top b \Leftrightarrow R^\top Q^\top QRx = R^\top Q^\top b \Leftrightarrow Rx = Q^\top b$$

El sistema final a resoldre

$$Rx = Q^\top b$$

és triangular superior i els valors $x = (x_1, x_2, \ldots, x_n)$ s'obtenen per *substitució cap a endarrere*.

El mètode que haureu de programar és una versió modificada de l'algorisme d'ortogonalització de Gram-Schmidt i té l'avantatge respecte d'aquest que la matriu Q obtinguda és de mida $m \times n$ enlloc de $m \times m$ i el nombre d'operacions emprades.

Problema: Donada una matriu $A \in \mathcal{M}_{m,n}$, calculeu la seva descomposició QR, amb Q ortogonal ($Q^\top Q = \mathrm{I}$) i R triangular superior.

Algorisme:

[0] T. Lázaro, M. Ollé i J.R. Pacha. Departament de Matemàtica Aplicada I

Pas 0: Definiu $A_1 = A = \left(a_1^{(1)}\ a_2^{(1)}\ \cdots a_n^{(1)}\right)$ on $a_j^{(1)}$ indica la columna j-ésima de la matriu A_1.

Pas 1: Normalitzem la columna $a_1^{(1)}$:

$$r_{11} = \|a_1^{(1)}\|, \qquad q_1 = \frac{a_1^{(1)}}{r_{11}}.$$

Ortogonalitzem respecte a aquesta columna totes les columnes posteriors:

$$r_{1s} = q_1^\top a_s^{(1)}, \qquad a_s^{(2)} = a_s^{(1)} - r_{1s} q_1.$$

Obtenim així la matriu $A_2 = \left(q_1\ a_2^{(2)}\ a_3^{(2)}\ \cdots a_n^{(2)}\right)$.

Pas k: Suposem que tenim ara la matriu $A_k = \left(q_1\ q_2\ \cdots q_{k-1}\ a_k^{(k)}\ a_{k+1}^{(k)}\ \cdots a_n^{(k)}\right)$. Per construcció, les seves columnes satisfan

$$q_j^\top q_\ell = \delta_{j\ell}, \qquad q_j^\top a_s^{(k)} = 0, \qquad j, \ell = 1 \div k-1, \quad s = k \div n.$$

Normalitzem la columna k-ésima,

$$r_{kk} = \|a_k^{(k)}\|, \qquad q_k = \frac{a_k^{(k)}}{r_{kk}},$$

i ortogonalitzem respecte de q_k les columnes $a_{k+1}^{(k)}, \cdots, a_n^{(k)}$,

$$r_{ks} = q_k^\top a_s^{(k)}, \qquad a_s^{(k+1)} = a_s^{(k)} - r_{ks} q_k, \qquad s = k+1 \div n.$$

S'obté d'aquesta manera $A_{k+1} = \left(q_1\ q_2\ \cdots q_k\ a_{k+1}^{(k+1)}\ a_{k+2}^{(k+1)}\ \cdots a_n^{(k+1)}\right)$ que verifica $q_j^\top q_\ell = \delta_{j\ell}$ i $q_j^\top a_s^{(k)} = 0$ per a $j, \ell = 1 \div k$, $s = k+1 \div n$.

Pas n: Després de n iteracions obtenim $A_{n+1} = (q_1\ q_2\ \cdots q_n)$, matriu ortogonal satisfent que

$$A = QR, \qquad Q = A_{n+1}, \qquad R = (r_{ks}).$$

Recordeu que si A és regular i totes les $r_{kk} > 0$ – com al nostre cas – la descomposició QR és única.

Exercici 2.1 *Programeu la descomposició QR d'una matriu A seguint aquest mètode. Useu-la per a calcular A^{-1}, $\mu_2(A)$ i $\mu_\infty(A)$, on $\mu_p(A) = \|A\|_p\ \|A^{-1}\|_p$ és el* número de condició *(en norma sub-p) de la matriu A.*

Exercici 2.2 *Les xifres oficials corresponents al padró municipal de Catalunya (vegeu la pàgina web de l'Institut d'Estadística de Catalunya IDESCAT) entre els anys 2000 i 2010 són aquestes:*

2000	*2001*	*2002*	*2003*	*2004*	*2005*
6.261.999	*6.361.365*	*6.506.440*	*6.704.146*	*6.813.319*	*6.995.206*

2006	*2007*	*2008*	*2009*	*2010*
7.134.697	*7.210.508*	*7.364.078*	*7.475.420*	*7.512.381*

Aleshores,

(*i*) *Busqueu el polinomi del grau escaient que millor aproximi aquesta taula en norma sub-2. Justifiqueu l'elecció mostrant una fita de l'error d'aquest i de la resta amb els que heu provat.*

(*ii*) *Useu-lo per a extrapolar la població de Catalunya l'any 2011 i compareu-la amb la xifra oficial (7.535.251). Quina seria la predicció pel 2012 ?*

(*iii*) *Feu una gràfica on apareguin els punts de la taula (representants per una rodona) i les gràfiques de les tres millors aproximacions.*

(*iv*) *Considereu el cas de* $p_{10}(x)$*, polinomi de grau* 10 *obtingut per mínims quadrats. Representeu-lo juntament amb els punts de la taula (aquests amb una rodona). Què observeu? Comenteu-lo.*

Comentari 2.3 *Numèricament és millor que considereu la taula anterior amb abscisses* $0, 1, \ldots, 10$ *enlloc de* $2000, 2001, \ldots, 2010$*.*

Un segon mètode, equivalent per a resoldre un sistema lineal sobredeterminat i per a trobar la projecció ortogonal d'una funció f sobre un espai de funcions $\mathcal{F}$, és trobar una base de polinomis ortogonals. En el cas de l'exercici 2.2, les abscisses són equidistants i, llavors, podem fer servir els anomenats *polinomis de Gram*, $P_{j,m}(t)$. Aquesta família de polinomis és ortogonal respecte de la taula $t_k = k$, $k = \div m$. Es poden obtenir a través de la següent recurrència:

$$\begin{aligned} P_{0,m}(t) &= 1, \qquad P_{1,m}(t) = 1 - \frac{2t}{m}, \\ (j+1)(m-j)P_{j+1,m}(t) &= (2j+1)(m-2t)P_{j,m}(t) - j(m+j+1)P_{j-1,m}(t), \qquad j = 1 \div m-1. \end{aligned}$$

En el cas que les nostres $m+1$ abscisses x_k estiguessin equidistribuïdes a l'interval $[a, b]$, caldria considerar el canvi de variable

$$x = a + \frac{b-a}{m}t \Leftrightarrow t = \frac{x-a}{b-a}m$$

i treballar amb els polinomis ortogonals $\tilde{P}_{j,m}(x) = P_{j,m}\left(\frac{x-a}{b-a}m\right)$

Exercici 2.4 (Continuació de l'exer. 2.2) *Considereu novament el problema de buscar un model per a la població catalana a la darrera dècada.*

(*i*) *Amb l'ajut d'un manipulador algebraic (Maple, Matlab, Octave, ...) calculeu els polinomis de Gram $P_{j,m}(t)$ associats a les abscisses $0, 1, \ldots, 10$ (que correspon al nostre cas, vegeu Comentari 2.3). Guardeu-los en un programa* `C` *o* `C++`*, conservant únicament els seus coeficients.*

(*ii*) *Plantegeu la resolució del problema 2.2 fent servir aquesta família de polinomis i responeu a les mateixes preguntes. Compareu els resultats obtinguts en ambdós casos. Recordeu que si necessiteu avaluar un polinomi en un cert punt una manera eficient de fer-ho és fent servir la Regla de Horner.*

Bibliografia

[1] A. Aubanell, A. Benseny, A. Delshams. *Eines bàsiques de càlcul numèric*. Universitat Autònoma de Barcelona, Barcelona, 1991.

[2] C. Bonet, À. Jorba, T.M-Seara, J. Masdemont, M. Ollé, A. Susin, M. Valencia. *Càlcul numèric*. Aula Teòrica 23, Edicions UPC, Barcelona, 1994.

[3] G. Dahlquist, A. Björck. *Numerical methods*. Prentice Hall Inc., New Jersey, 1974.

[4] A. Quarteroni, F. Saleri. *Scientific Computing with MATLAB and Octave, Second Edition*. Springer-Verlag, Berlín, Heidelberg, 2006.

[5] J. Stoer, R. Bulirsch. *Introduction to numerical analysis* (third edition). Texts in Applied Mathematics, Springer-Verlag, New York - Berlin - Heidelberg, 2002.

3 Integració Numèrica

L'objectiu d'aquesta pràctica és calcular aproximacions per a una integral donada fent servir diferents mètodes d'integració numèrica. Així doncs, considereu

$$I = \int_0^1 \mathrm{e}^{-x^2}\, dx.$$

Exercici 3.1 *Obteniu una aproximació del valor d'I amb* 6 *xifres decimals correctes fent servir la Regla dels Trapezis composta. Justifiqueu analíticament el nombre N de subintervals que es necessiten per aconseguir-ho.*

Considereu ara l'aproximació d'I obtinguda utilitzant precisió simple i precisió doble amb $N =$ 800 *intervals. Compareu els resultats obtinguts i justifiqueu-los.*

Exercici 3.2 *Obteniu una aproximació d'I amb la mateixa precisió que a l'apartat anterior fent servir la Regla de Simpson composta. Amb quants intervals N caldrà treballar en aquest cas?*

Recordeu que, a partir de la *F*órmula d'Euler-Mc Laurin i suposant $f \in \mathcal{C}^{2s+2}([a,b])$, és possible obtenir el següent desenvolupament asimptòtic de l'error per a la Regla dels Trapezis:

$$T(h) = \int_a^b f(x)\, dx + \sum_{r=1}^{s} h^{2r} \frac{B_{2r}}{(2r)!}\left[f^{(2r-1)}(b) - f^{(2r-1)}(a)\right] + \frac{B_{2s+2}}{(2s+2)!}(b-a) f^{(2s+2)}(\xi)\, h^{2s+2}, \tag{1}$$

a on $\xi \in (a,b)$ i els B_j són els anomenats *nombres (clàssics) de Bernoulli*. Aquests nombres es poden definir a partir del desenvolupament

$$\frac{x}{e^x - 1} = B_0 + \frac{B_1}{1!}x + \frac{B_2}{2!}x^2 + \frac{B_3}{3!}x^3 + \cdots = \sum_{n\geq 0} \frac{B_n}{n!}x^n,$$

i obtenir-se a gràcies a la recurrència

$$B_0 = 1, \quad B_1 = -\frac{1}{2}, \qquad \sum_{\ell=0}^{j-1} \binom{j}{\ell} B_\ell = 0, \quad j \geq 3.$$

Per exemple, els primers nombres de Bernoulli són

$$B_0 = 1, \quad B_1 = -1/2, \quad B_2 = 1/6, \quad B_4 = -1/30, \quad B_6 = 1/42, \quad B_8 = -1/30$$

[0]T. Lázaro, M. Ollé i J. R. Pacha. Departament Matemàtica Aplicada I

i es pot demostrar que $B_{2k+1} = 0$ per a $k \geq 1$.

A partir de l'expressió asimptòtica donada a (1), podem considerar les anomenades *fórmules* corregides de la Regla dels Trapezis, consistents en agafar alguns dels primers termes del desenvolupament d'Euler-Mc Laurin. Concretament, si prenem $s = 1$ a la fórmula (1) obtenim

$$\int_a^b f(x)\,dx = C_1T(h) + \frac{1}{30 \cdot 4!}(b-a)f^{(4)}(\xi)h^4, \qquad \xi \in (a,b),$$

a on

$$C_1T(h) = T(h) + \frac{h^2}{12}\left[f'(a) - f'(b)\right].$$

De manera anàloga, prenent $s = 2$ s'obté:

$$\int_a^b f(x)\,dx = C_2T(h) - \frac{1}{42 \cdot 6!}(b-a)f^{(6)}(\xi)h^6, \qquad \xi \in (a,b),$$

amb

$$C_2T(h) = T(h) + \frac{h^2}{12}\left[f'(a) - f'(b)\right] - \frac{h^4}{30 \cdot 4!}\left[f^{(3)}(a) - f^{(3)}(b)\right].$$

Exercici 3.3 *Obteniu aproximacions del valor d'I amb un error absolut inferior a* $0.5 \cdot 10^{-6}$*, fent servir les fórmules de Trapezis corregides* $C_1T(h)$ *i* $C_2T(h)$*. Quants subintervals N heu escollit en cada cas ? Justifiqueu la resposta.*

Exercici 3.4 *Apliqueu el mètode d'*extrapolació de Romberg *per calcular I amb un error relatiu inferior a* $0.5 \cdot 10^{-7}$*. Construïu l'esquema triangular*

$$\begin{array}{lllll} T_{00} & & & & \\ T_{10} & T_{11} & & & \\ T_{20} & T_{21} & T_{22} & & \\ \vdots & \vdots & \vdots & \ddots & \\ T_{i0} & T_{i1} & T_{i2} & \cdots & T_{ij} \end{array}$$

on T_{i0} *correspon al valor obtingut per la Regla dels Trapezis amb* 2^i *subintervals. Useu com a condició d'aturada del procés que la següent estimació de l'error relatiu es satisfaci:*

$$\left|\frac{T_{i,j} - T_{i,j-1}}{T_{i0}}\right| < \frac{1}{2} \cdot 10^{-7}$$

4 Descomposició en Valors Singulars (SVD)

La descomposició QR d'una matriu és un mètode comú per a resoldre sistemes sobredeterminats i per trobar solucions a problemes de mínims quadrats, però no és pas l'únic. Un altre molt comú, i amb gran quantitat d'aplicacions, és l'anomenada *Descomposició en Valors Singulars* d'una matriu o, en anglès, *Singular Value Decomposition* (SVD).

Teorema 4.1 (SVD [2])**.** *Sigui $A \in \mathcal{M}_{m,n}(\mathbb{R})$, amb $m \geq n$. Aleshores, existeix una descomposició de la matriu A de la forma*

$$A = U\Sigma V^\top \tag{1}$$

satisfent les següents propietats:

- *$U \in \mathcal{M}_{m,n}(\mathbb{R})$ és matriu ortogonal, $U^\top U = I_n$. Els vectors-columna que formen U, és a dir $U = (u_1\ u_2\ \cdots\ u_n)$, reben el nom de* vector singulars per l'esquerra.
- *$V \in \mathcal{M}_{n,n}(\mathbb{R})$ és també ortogonal, $V^\top V = I_n$. Anàlogament al punt anterior, els vectors-columna que formen V, o sigui $V = (v_1\ v_2 \cdots\ v_n)$, s'anomenen* vectors singulars per la dreta.
- *$\Sigma \in \mathcal{M}_{n,n}(\mathbb{R}) = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ amb $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$, anomenats* valors singulars *de la matriu A. El valor σ_1 es diu* valor singular principal.

Comentari 4.2.

1. *En el cas $A \in \mathcal{M}_{m,n}(\mathbb{R})$ amb $m < n$ el mateix teorema s'aplica. És trivial comprovar que la descomposició SVD d'$A^\top$ és $A^\top = V\Sigma^\top U^\top$.*
2. *Per a matrius complexes el teorema és també cert canviant "trasposada" per "Hermítica": $A = U\Sigma V^*$ amb $U^*U = V^*V = I_n$.*

Demostració del Teorema 4.1. Definim la matriu simètrica $S := A^\top A \in \mathcal{M}_{n,n}(\mathbb{R})$. Pel teorema espectral, existeix una base ortonormal de $\mathbb{R}^n$ que consisteix en els VEPs de S i, a més, els VAPs corresponents a cada vector són reals; i.e., existeixen n vectors $v_1, v_2, \ldots, v_n \in \mathbb{R}^n$ t.q.:

$$Sv_i = \lambda_i v_i,$$

amb $\mathbb{R}^n = [v_1, v_2, \ldots, v_n]$ i $\lambda_i \in \mathbb{R}$, $\langle v_i, v_j \rangle = \delta_{i,j}$ per tot $i, j = 1 \div n$. En aquest cas, de fet, $\lambda_i \geq 0$ per tot $i = 1 \div n$, ja que S és semidefinida positiva. En efecte, sigui $\xi \in \mathbb{R}^n$, $\xi \neq 0$, llavors

$$\langle \xi, S\xi \rangle = \langle \xi, A^\top A\xi \rangle = \langle A\xi, A\xi \rangle = \|A\xi\|_2^2 \geq 0.$$

Sense pèrdua de generalitat, podem suposar que els vectors de la base ortonormal vénen ordenats de manera que els vaps respectius són decreixents, i.e.: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Sigui $1 \leq r \leq n$

[0]T. Lázaro, M. Ollé i J.R. Pacha. Departament Matemàtica Aplicada I

t.q.: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0$, $\lambda_{r+1} = \lambda_{r+2} = \cdots = \lambda_n = 0$ (notem que pot ser $r = n$). A continuació definim

$$\sigma_1 := \sqrt{\lambda_1}, \sigma_2 := \sqrt{\lambda_2}, \ldots, \sigma_r := \sqrt{\lambda_r} \tag{2}$$

de manera que: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Introduïm també els vectors

$$u_1 = \frac{1}{\sigma_1} A v_1, u_2 = \frac{1}{\sigma_2} A v_2, \ldots, u_r = \frac{1}{\sigma_r} A v_r.$$

Aquest vectors són ortogonals dos a dos i unitaris. Comprovem-ho:

$$\langle u_i, u_j \rangle = \frac{1}{\sigma_i \sigma_j} \langle A v_i, A v_j \rangle = \frac{1}{\sigma_i \sigma_j} \langle v_i, A^\top A v_j \rangle = \frac{1}{\sigma_i \sigma_j} \langle v_i, S v_j \rangle = \frac{\lambda_j}{\sigma_i \sigma_j} \langle v_i, v_j \rangle = \frac{\sigma_j}{\sigma_i} \langle v_i, v_j \rangle = \delta_{ij}$$

per $i, j = 1 \div r$, on hem fet servir que, d'acord amb (2) és $\lambda_j = \sigma_j^2$, per $j = 1 \div r$. A continuació, completem els vectors $u_1, u_2, \ldots, u_r \in \mathbb{R}^m$ amb $n - r$ vectors $u_{r+1}, u_{r+2}, \ldots, u_n \in \mathbb{R}^n$ de manera que $\{u_1, u_2, \ldots, u_n\}$ formin una família ortonormal a $\mathbb{R}^m$. Ara, en la base ortonormal $\{v_1, v_2, \ldots, v_n\}$ de $\mathbb{R}^n$ i per a la família ortonormal $\{u_1, u_2, \ldots, u_n\}$ de $\mathbb{R}^m$, tenim

$$\begin{aligned} A v_j &= \sigma_j \frac{1}{\sigma_j} A v_j = \sigma_j u_j, && \text{per } j = 1 \div r, \\ A v_j &= 0, && \text{per } j = r+1 \div n \end{aligned} \tag{3}$$

d'acord amb la definició de u_j, $j = 1 \div r$. Ara, si escrivim $U = \begin{pmatrix} u_1 & u_2 \cdots u_n \end{pmatrix} \in \mathcal{M}_{m,n}(\mathbb{R})$ com la matriu que té per columnes els vectors $u_1, u_2, \ldots, u_n$; $V = \begin{pmatrix} v_1 & v_2 \cdots v_n \end{pmatrix} \in \mathcal{M}_{n,n}(\mathbb{R})$ com la matriu que té per columnes els vectors $v_1, v_2, \ldots, v_n$ i $\Sigma = \operatorname{diag}[\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0] \in \mathcal{M}_{n,n}(\mathbb{R})$; podem expressar (3) en forma matricial com

$$AV = U\Sigma. \tag{4}$$

Finalment, com que les columnes de V són ortogonals, tenim que $V^\top V = V V^\top = I_n$, i multiplicant (4) per la dreta a totes dues bandes per $V^\top$ s'obté la descomposició buscada. □

La descomposició SVD d'una matriu satisfà moltes propietats importants. Al següent Teorema donem les més destacades.

Teorema 4.3 ([2]). *Sigui $A = U\Sigma V^\top$ la descomposició SVD de la matriu $A \in \mathcal{M}_{m,n}(\mathbb{R})$, amb $m \geq n$. Aleshores:*

1. *Suposeu que A és simètrica i que, per tant, admet una descomposició de la forma $A = UDU^\top$ amb $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $U = (u_1 \ u_2 \ \cdots \ u_n)$ ortogonal, és a dir $UU^\top = I_n$. Aleshores una SVD d'A és $A = U\Sigma V^\top$ amb $\sigma_i = |\lambda_i|$ i $v_i = \operatorname{sign}(\lambda_i) u_i$, on* sign *és la funció "signe" i* $\operatorname{sign}(0) = 1$.

2. *Els vaps de la matriu simètrica $A^\top A \in \mathcal{M}_{n,n}(\mathbb{R})$ són σ_i^2. Els vectors singulars per la dreta $\{v_1, v_2, \ldots, v_n\}$ formen una base ortogonormal de veps.*

3. *Els vaps de la matriu simètrica* $AA^\top \in \mathcal{M}_{m,m}(\mathbb{R})$ *són* σ_i^2, $i = 1, 2, \ldots, n$, *i* $m - n$ *zeros. Els vectors singulars per l'esquerra* u_i *són els vep's ortonormals corresponents als vaps* σ_i^2.

4. *Siguin* $A \in \mathcal{M}_{n-n}(\mathbb{R})$ *una matriu quadrada i* $A = U\Sigma V^\top$ *la seva* SVD *amb* $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n)$, $U = (u_1\ u_2\ \cdots\ u_n)$ *i* $V = (v_1\ v_2\ \cdots\ v_n)$. *Considereu la matriu per blocs*

$$H = \begin{pmatrix} 0 & A^\top \\ A & 0 \end{pmatrix}.$$

Aleshores, els vaps d'H són $\{\pm\sigma_1, \pm\sigma_2, \ldots, \pm\sigma_n\}$ *i els seus veps unitaris corresponents són*

$$\frac{1}{\sqrt{2}} \begin{pmatrix} v_i \\ \pm u_i \end{pmatrix}.$$

5. *Si A té rang màxim, és a dir,* $\mathrm{rang}(A) = n$, *llavors*

$$\min_x \|Ax - b\| = \|Ax^* - b\|$$

amb $x^* = V\Sigma^{-1}U^\top b$. *Per tant, proporciona la solució del problema de mínims quadrats.*

6. *La norma euclídea d'A és* $\|A\|_2 = \sigma_1$. *Si* $A \in \mathcal{M}_{n,n}(\mathbb{R})$ *és no singular aleshores* $\|A^{-1}\|_2 = 1/\sigma_n$ *i el nombre de condició* $\mu_2(A) = \|A\|_2\|A^{-1}\|_2 = \sigma_1/\sigma_n$.

7. *Suposem* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_n = 0$. *LLavors,* $\mathrm{rang}(A) = r$ *i*

$$\ker A = \mathrm{span}\{v_{r+1}, v_{r+2}, \ldots, v_n\}, \qquad \mathrm{Im} A = \mathrm{span}\{u_1, u_2, \ldots, u_r\}.$$

8. *Siguin* $S_n = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ *l'esfera unitat n-dimensional i* $A(S_n) = \{Ax : x \in S_n\}$ *la seva imatge per la matriu* $A \in \mathcal{M}_{m,n}(\mathbb{R})$. *Aleshores,* $A(S_n)$ *és un el·lipsoide de centre l'origen d'*$\mathbb{R}^m$ *i eixos principals* $\sigma_i u_i$.

Demostració. Cadascun dels apartats de la proposició es proven a partir del Teorema 4.1. Deixem doncs la demostració d'aquest teorema com exercici (o bé podeu consultar la referència [2]). *Nota:* aquesta activitat és voluntària i *no* cal que la presenteu a l'informe de la pràctica. □

Exercici 4.4. *(Obligatori, cal que figuri a l'informe). Torneu a fer l'exercici 2 de la 2ª pràctica (extrapolació de la població a Catalunya) aplicant —en fer l'aproximaxió per mínims quadrats—, la propietat 5 del Teorema 4.3 (en comptes de la descomposició QR). Compareu els resultats.*

Una conseqüència molt interessant del Teorema 4.1 és que proporciona un mètode d'aproximació de la matriu A per matrius de menor rang. De fet,

Proposition 4.5 ([2]). *Considereu la descomposició SVD d'una matriu $A \in \mathcal{M}_{m,n}(\mathbb{R})$, $A = U\Sigma V^\top$ i escriviu $U = [u_1\ u_2\ \dots u_n]$ i $V = [v_1\ v_2\ \dots v_n]$, on cada u_j, v_j és un vector columna d'm components. Observeu que podem escriure, equivalentment, que*

$$A = U\Sigma V^\top = \sum_{i=1}^{n} \sigma_i u_i v_i^\top$$

essent $\{\sigma_1, \sigma_2, \dots, \sigma_r, \sigma_{r+1} = 0, \dots, \sigma_n = 0\}$ els valors singulars d'A. Aleshores,

$$A_k := \sum_{i=1}^{k} \sigma_i u_i v_i^\top = U\Sigma_k V^\top,$$

amb $\Sigma_k = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_k, 0, \dots, 0)$, és la matriu de rang k que millor aproxima A en norma euclídea. A més, $\|A - A_k\|_2 = \sigma_{k+1}$.

Demostració. Es comprova fàcilment que $\|A - A_k\|_2 = \sigma_{k+1}$. Sigui $B \in \mathcal{M}_{m,n}(\mathbb{R})$ de rang $k < n$ i, per tant, amb $\dim \mathrm{Nuc}B = n - k$. Considerem el subespai $F = [v_1, v_2, \dots, v_{k+1}]$. Aquests subespais, F i Nuc$B$, es solapen, ja que la suma de les seves dimensions $(k+1) + (n-k) > n$. Sigui ξ un vector de la seva intersecció, que agafarem unitari; és a dir, t.q.: $\|\xi\|_2 = 1$. Llavors,

$$\|A - B\|_2^2 \geq \|(A-B)\xi\|_2^2 = \|A\xi\|_2^2 = \|U\Sigma V^\top \xi\|_2^2 = \|\Sigma V^\top \xi\|_2^2 \geq \sigma_{k+1}^2 \|V^\top \xi\|_2^2 = \sigma_{k+1}^2.$$

La qual cosa completa la prova. □

Una aplicació divertida a la vegada que pràctica d'aquest darrer resultat és la compresió d'imatges. L'il·lustrarem amb un exemple complet. Suposeu que teniu una fotografia en format `jpg` com aquesta que anomenem `camell.jpg`. El primer que farem serà convertir-la a escala de grisos i

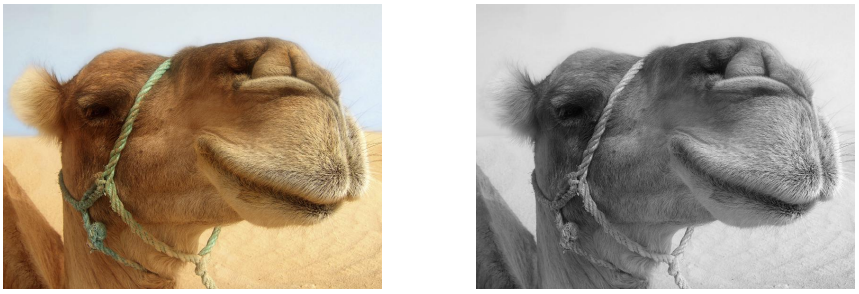

Figura 1: Retrat d'un camell (color i blanc i negre).

després transformar-la en una matriu en doble precisió segons la intensitat de cada píxel o grup de píxels. Malgrat es pot fer amb programes de tractament d'imatge de lliure distribució (per exemple, l'`opencv`) nosaltres usarem `Matlab`. Les comandes són aquestes:

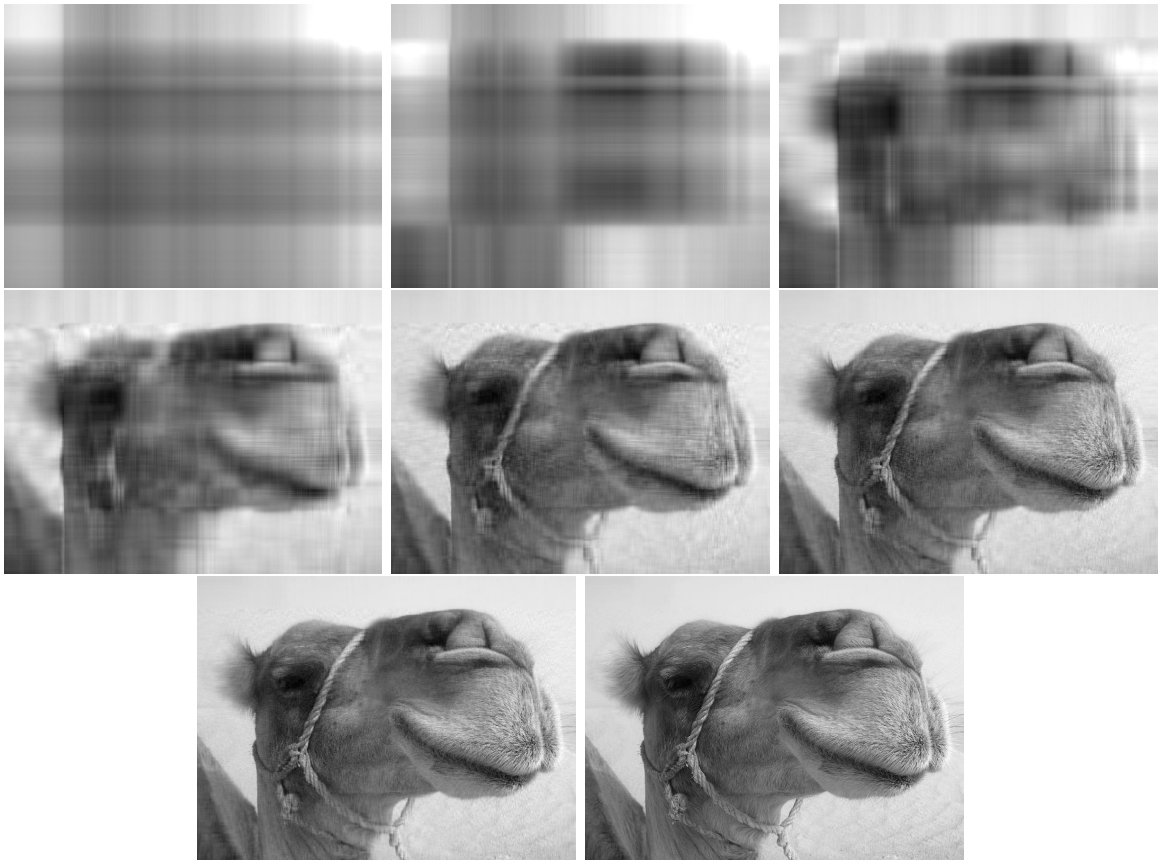

Figura 2: De dalt a baix i d'esquerra a dreta: fotografies corresponents a les diferents aproximacions usant els primers $1, 2, 5, 10, 25, 50, 100, 298$ valors singulars d'A, respectivament.

```
A = imread('camel.jpg');
A = rgb2gray(A);
A = im2double(A);
imshow(A)
dlmwrite('camel_matrix.dat', A, 'delimiter', '\t', 'precision', 4)
size(A)
```

La comanda `imshow(A)` mostra la imatge de nou mentre que `dlmwrite` escriu la matriu d'intensitats A en un fitxer ascii anomenat `camel_matrix.dat` de 600 files i 800 columnes separades per una tabulació. Si tornem a anomenar A la matriu (numèrica, d'intensitats) guardada al fitxer `camel_matrix.dat` i l'apliquem la SVD , obtenim les matrius que l'aproximen $A_1, A_2, A_5, A_{10}, A_{25}, A_{100}$

i A_{298}, obtingudes considerant els $1, 2, \ldots, 298$ primers valors no singulars d'A, respectivament. Si volem recuperar la imatge que correspon a cada una d'aquestes aproximacions, farem a MATLAB:

```
imwrite(Ak, sprintf('camel-%d.jpg', k));
```

on $k = 1, 2, \ldots, 298$. El resultat el podeu veure a la Figura 2.

Observeu que ja amb només 100 valors singulars (penúltima fotografia a Figura 2) la resolució és força raonable.

Exercici 4.6. *Feu el mateix amb una fotografia de vosaltres (és a dir, dels dos components de cada grup). És important que preneu la fotografia a una resolució baixa per a evitar que la matriu surti massa gran. L'exemple del camell está fet a una resolució de* 600×800.

5 Presentació d'informes

Els informes s'hauran de lliurar en format PDF a la Intranet (Campus Digital Atenea), a l'espai que, al llarg de la setmana del 17 al 22 de desembre i amb data límit de divendres 22 a les 24 hores estarà actiu per pujar fitxers.

Cada informe haurà de contenir la resolució comentada dels exercicis 4.4 i 4.6 plantejats en aquest guió, una descripció detallada de l'algorisme triat per programar la descomposició SVD, així com la bibliografia (llibres, articles, pàgines güeb,...) que heu consultat.

Referències

[1] G. Dahlquist, A. Björck. *Numerical methods in Scientific Computing, vol 1, 2.* SIAM, Philadelphia, 2008.

[2] J. Demmel *Applied Numerical Linear Algebra*. SIAM, Philadelphia, 1997. 1, 2, 3, 4

[3] Wikipedia: http://en.wikipedia.org/wiki/Singular_value_decomposition

5 Introducció a la resolució numèrica d'edos

El propòsit d'aquestes notes és introduir l'alumne en alguns del problemes bàsics que hom es troba quan vol simular numèricament molts sistemes dinàmics.

No dedicarem temps a la presentació i programació (que sempre resta com a possibilitat per a l'estudiant que ho dessitgi) dels mètodes clàssics d'integració numèrica d'edos (d'un pas o multipas) sino que farem servir directament un integrador, Taylor, dissenyat per n'À. Jorba (Universitat de Barcelona) i M- Zou (University of texas). Aquest software, de lliure accés, amb codi obert i pensat per a nivell de recerca més que no pas de docència, ha demostrat els darrers anys un nivell molt d'alt de competitivitat (tenint en compte que no deixa de ser general i que hom pot crear sempre integradors més eficients *ad-hoc* per a un sistema concret) en quant a velocitat i precisió. Els algorismes que presentem en aquestes pàgines han estat directament estrets d'un article (un clàssic) d'en C. Simó [3] a on es mostra un ventall d'eines numèriques i analítiques essencials per a afrontar molts problemes en sistemes dinàmics.

5.1 El mètode de Taylor

Els darrers anys hi ha hagut un desenvolupament de programes que generen de manera automàtica rutines d'integració numèrica d'edos fent servir el mètode de Taylor, posant de manifest la seva potència computacional en comparació amb altres mètodes (Runge-Kutta, Adams-Bashforth, ...). Un d'aquests programes és l'anomenat Taylor, creat pels professors À. Jorba (Universitat de Barcelona) i M. Zou (University of Texas). El podeu descarregar des de l'adreça (Taylor-home). En el cas que no estigués instal-lat al pc de l'aula o volguéssim instalar-lo a qualsevol altre ordinador seguiríem les instruccions indicades allà.

Seguirem la documentació de Taylor [1, 2], centrant-nos en un primer contacte amb ell i deixant per a l'estudiant interessat una lectura més profunda. Per facilitar-ne la comprensió ho aplicarem al càlcul (i dibuix) d'una òrbita d'un sistema concret. En els nostre cas, hem escollit **l'equació de Lorenz**.

El sistema de Lorenz és un exemple de sistema 3D amb comportament caòtic:

$$\left\{ \begin{array}{rcl} \dot{x} & = & \sigma(y-x) \\ \dot{y} & = & \rho x - y - xz \\ \dot{z} & = & xy - \beta z. \end{array} \right. \tag{1}$$

És ben coneguda l'existència d'una atractor per a determinats valors dels seus paràmetres: $\sigma = 10$, $\rho = 28$, $\beta = 8/3$). Aquest atractor va ser una de les primeres mostres de l'anomenat *butterfly effect* (efecte papallona). Podeu trobar fàcilment a internet vídeos que el mostrin. Una ràpida tria podria ser, per exemple, aquesta: video-1, video-2, video-3.

[0] T. Lázaro, J. R. Pacha i M. Ollé. Departament Matemàtica Aplicada I

Usarem Taylor per integrar-ho numèricament. El primer que hem de fer és definir un programa amb el cap de vectors. L'anomenarem lorenz.eq1 i constarà de les següents línies:

```
RR=28.0;
diff(x,t)=10.0*(y-x);
diff(y,t)=RR * x - y - x*z;
diff(z,t)=x*y -(8.0/3.0)*z;
```

Graveu-lo a la vostra carpeta de treball. Ara hem de cridar a Taylor per a que creï el programa que efectuarà un pas simple d'integració numèrica del nostre sistema. Per fer-ho, escrivim:

```
perico@delospalotes: taylor -name lrnz -o lorenz.c -jet -step lorenz.eq1
perico@delospalotes: taylor -name lrnz -o taylor.h -header
```

La primera línia crea el fitxer `lorenz.c`, que conté el codi que calcula les derivades necessàries fins a l'ordre convenient del nostre camp (l'anomenat *jet of derivatives*, amb l'opció `-jet`) i el control de pas (`-step`). L'opció `-name` indica a Taylor que afegeixi el suffix `lrnz` al nom del fitxer resultant, `taylor_step_lrnz`. La segona línia produeix el *header file* `taylor.h`, necessari per compilar `lorenz.c`. Un cop executades aquestes comandes tindreu a la vostra carpeta els següents fitxers:

```
perico@delospalotes: ls
lorenz.c  lorenz.eq1  taylor.h
```

Tot i que Taylor també treballa amb precisió *estesa* nosaltres farem servir l'opció per defecte, la precisió doble del nostre ordinador. Podem cridar Taylor de diverses maneres:

5.1.1 Execució directa de Taylor

Taylor permet crear un programa simple que ens permet calcular els punts d'una òrbita concreta demanada per línia de comandes. Per fer-ho, escrivim

```
perico@delospalotes: taylor -name lrnz -o main_lrnz.c -main_only lorenz.eq1
```

el compilem

```
perico@delospalotes: gcc -O3 main_lrnz.c lorenz.c -lm -s
```

generant-se l'executable `a.out`. Si l'executem veurem que ens demana les condicions inicials de la nostra òrbita i les toleràncies desitjades:

```
perico@delospalotes: ./a.out
Enter Initial xx[0]: 0.1
Enter Initial xx[1]: 0.2
Enter Initial xx[2]: 0.3
```

```
Enter start time: 0.0
Enter stop time: 0.5
Enter absolute error tolerance: 0.1e-16
Enter relative error tolerance: 0.1e-16
```

La sortida per pantalla serà una taula amb els valors obtinguts $x(t), y(t), z(t), t$:

```
0.1 0.2 0.3 0
0.169464 0.36274 0.265313 0.0484829
0.309703 0.671984 0.236866 0.100502
0.539572 1.17353 0.225961 0.147831
0.88444 1.92431 0.245377 0.189871
1.38449 3.01108 0.322239 0.227996
2.0651 4.48536 0.503345 0.262065
2.96612 6.42411 0.86911 0.29303
4.11167 8.85801 1.53644 0.321165
5.50045 11.743 2.64885 0.346567
7.12416 14.9901 4.37751 0.369644
8.95632 18.4314 6.90055 0.39078
10.9434 21.7988 10.363 0.410287
13.0013 24.7287 14.8242 0.428432
15.005 26.7883 20.1692 0.445362
16.8064 27.5863 26.0763 0.46119
18.2881 26.8643 32.1801 0.476329
19.3086 24.5336 37.9112 0.490913
```

Observeu que la darrera dada que treu per pantalla correspon a temps `0.490913` i no pas `0.5`, com havíem seleccionat. Això és degut a que el càlcul del nou punt es fa dins d'un bucle `do/while` amb la condició de que el temps emprat en aquell moment no passi el valor marcat a `stop_time`. Per a obtenir el valor corresponent a temps `stop_time` (`0.5` en el nostre exemple) caldrà fer-li escriure un iterat més de l'integrador.

Si volguéssim representar-ho gràficament amb l'ajut del `gnuplot` podríem demanar que la sortida anés a un fitxer de dades:

```
perico@delospalotes: ./a.out > exemple-orbita-lrnz.dat
```

A més, per a que l'òrbita a representar sigui més llarga repetirem l'exemple anterior però amb `stop_time` igual a tt 30. Un cop fet això cridem a `gnuplot`:

```
perico@delospalotes$ gnuplot

        G N U P L O T
```

```
        Version 4.4 patchlevel 2
        last modified Wed Sep 22 12:10:34 PDT 2010
        System: Linux 2.6.38-9-generic-pae

        Copyright (C) 1986-1993, 1998, 2004, 2007-2010
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help seeking-assistance"
        immediate help:   type "help"
        plot window:      hit 'h'

Terminal type set to 'wxt'
gnuplot> set term postscript
Terminal type set to 'postscript'
Options are 'landscape noenhanced defaultplex \
   leveldefault monochrome colortext \
   dashed dashlength 1.0 linewidth 1.0 butt noclip \
   palfuncparam 2000,0.003 \
   "Helvetica" 14 '
gnuplot> set output "./exemple-orbita.ps"
gnuplot> splot "exemple-orbita-lrnz.dat" u 1:2:3 with lines
gnuplot> quit
perico@delospalotes:
```

Si obrim el fitxer `exemple-orbita.ps` amb l'`okular`, per exemple o qualsevol visualitzador de postscripts, obtenim:

Comentari 5.1 *Un manera alternativa de generar un petit programa principal* `lorenz.c` *que contingui el* header*, el codi de control de pas i el programa que generi el* jet *de derivades és fer*

```
perico@delospalotes: taylor -o lorenz.c lorenz.eq1
```

*Compilant-lo (*linkant*-hi la llibreria matemàtica):*

```
perico@delospalotes: gcc -O3 lorenz.c -lm
```

obtenim un fitxer executable `a.out` *com abans.*

5.1.2 Execució de Taylor dins d'un programa

Per fer-ho, cal inicialment demanar a Taylor que generi l'integrador i el fitxer *header* `taylor.h`:

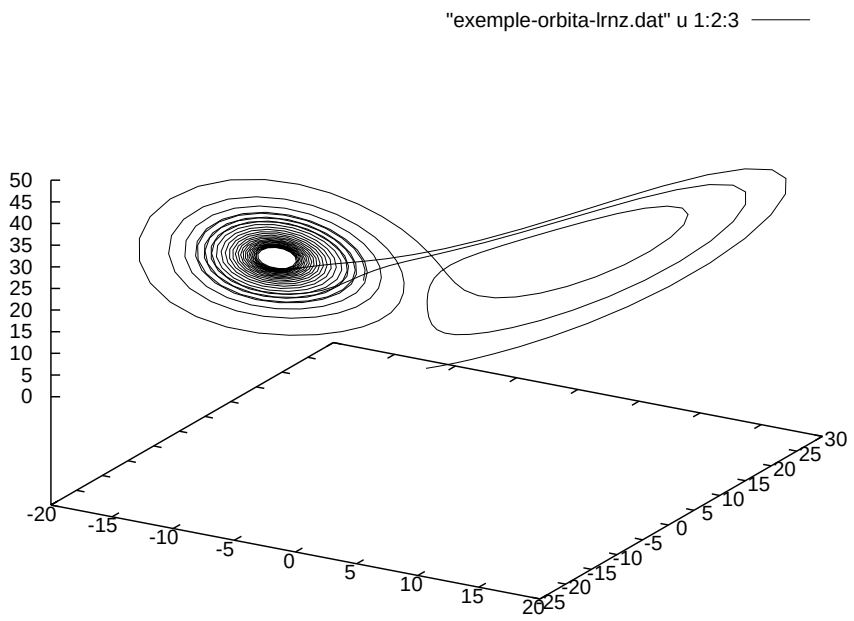

Figure 1: Una òrbita del sistema de Lorenz

```
perico@delospalotes: taylor -o lorenz.c -jet -step -name lorenz lorenz.eq1
perico@delospalotes: taylor -o taylor.h -header
```

Aquesta primera línia genera un fitxer `lorenz.c` que haurem de *linkar* i compilar juntament amb el fitxer que contingui l'algorisme de crida a l'integrador i que anomenarem, per exemple, `main1.c`:

```
#include "math.h"
#include "stdio.h"
#include "taylor.h"

#define max(a,b) ((a)<(b) ? (b) : (a))
#define sgn(a)   ((a)<0 ? -1 : 1 )
#define abs(a)   ((a)<0 ? -(a) : (a))

// Taylor control parameters
#define H_INICIAL 0.001
#define ABS_ERR 1.0e-16
#define REL_ERR 1.0e-16


int main(int argc, char **argv)
```

```
{
// xx[]: variables espaials,  t: temps
double xx[3], t;
// Constants al Taylor
double h, abs_err, rel_err, h_return, log10abs_err;
double log10rel_err, endtime;
int nsteps = 20, step_ctrl_method = 2, direction = 1;
int order = 10;
//
int nombre_passos; // nombre de passos d'integraci

// Fixem les condicions inicials
xx[0] = 0.1;
xx[1] = 0.2;
xx[2] = 0.3;
t = 0.0;

// Assignem parametres de control de Taylor
h= (double) H_INICIAL;
abs_err = (double) ABS_ERR ;
rel_err = (double) REL_ERR;
log10abs_err = log10(abs_err);
log10rel_err = log10(rel_err);

// ================================================================================
// Exemple 1:  integrem nombre_passos passos de l'edo fins a arribar, com a maxim,
// a temps endtime

endtime = 100.;

nombre_passos=20;
printf("\n Exemple 1: integrem la nostra edo %d passos  \n\n",nombre_passos);

h_return=h;
while( -- nsteps >= 0 && h_return != 0) {
  // Per exemple treiem els resultats per pantalla
  printf("x:%f  \ty:%f \tz:%f  \tt:%f  \th:%e \tpas:%d\n",
         xx[0],xx[1],xx[2],t,h_return,(nombre_passos-nsteps));

  taylor_step_lorenz(&t, &xx[0], direction,
```

```
  step_ctrl_method,log10abs_err, log10rel_err,
  &endtime, &h_return, &order);
  }

// ========================================================================
// Exemple 2:  integrem fins a arribar a endtime

endtime = 2.0;
printf("\n\n\n\n Exemple 2: integrem la nostra edo fins a temps %g  \n\n",endtime);
// Tornem a les condicions inicials
xx[0] = 0.1;
xx[1] = 0.2;
xx[2] = 0.3;
t = 0.0;
h= (double) H_INICIAL;

h_return=h;
while( (t<endtime) && h_return != 0) {
  // Per exemple treiem els resultats per pantalla
  printf("x:%f  \ty:%f \tz:%f  \tt:%f \th:%e\n", xx[0],xx[1],xx[2],t,h_return);

  taylor_step_lorenz(&t, &xx[0], direction,
  step_ctrl_method,log10abs_err, log10rel_err,
  &endtime, &h_return, &order);
  }
printf("x:%f  \ty:%f \tz:%f  \tt:%f \th:%e\n", xx[0],xx[1],xx[2],t,h_return);

return(0);
}
```

Un cop gravat aquest fitxer `main1.c`, els linkem i compilem, generant el nostre executable `IntLorenz`:

```
perico@delospalotes: gcc -o IntLorenz lorenz.c main1.c -lm
```

Recordeu que si no especifiquem l'executable de sortida, per defecte el nom que dona `gcc` és `a.out`. Per executar el programa només cal fer

```
perico@delospalotes: ./IntLorenz
```

i sortirà per pantalla una taula similar a l'obtinguda a la secció anterior.

Taylor també permet modificar des del fitxer principal els paràmetres que puguin aparèixer en el nostre camp vectorial. Fem-ho amb l'exemple lorenz.eq1 que abans teníem escrit així:

```
RR=28.0;
diff(x,t)=10.0*(y-x);
diff(y,t)=RR * x - y - x*z;
diff(z,t)=x*y -(8.0/3.0)*z;
```

Caldrà ara definir les variables `RR` i `coef[i]`, $i = 0, 1, 2$ com a externes:

```
extern MY_FLOAT RR, coef[3];
diff(x,t)=coef[0]*(y-x);
diff(y,t)=RR * x - y - x*z;
diff(z,t)=x*y -(coef[1]/coef[2])*z;
```

I al nostre programa principal main1.c haurem d'escriure

```
#include "math.h"
#include "stdio.h"
#include "taylor.h"

#define max(a,b) ((a)<(b) ? (b) : (a))
#define sgn(a)   ((a)<0 ? -1 : 1 )
#define abs(a)   ((a)<0 ? -(a) : (a))

// Taylor control parameters
#define H_INICIAL 0.001
#define ABS_ERR 1.0e-16
#define REL_ERR 1.0e-16

// Variables externes. Parametres del camp vectorial.
MY_FLOAT RR, coef[3];

int main(int argc, char **argv)
{
// xx[]: variables espaials,  t: temps
double xx[3], t;
// Constants al Taylor
double h, abs_err, rel_err, h_return, log10abs_err;
double log10rel_err, endtime;
int nsteps = 20, step_ctrl_method = 2, direction = 1;
int order = 10;
.....
RR=28.0;
```

```
coef[0]=10.0; coef[1]=8.0; coef[2]=3.0;
.....
```

Trobareu més informació sobre les diverses opcions possibles al manual de Taylor [1].

5.2 Càlcul de l'aplicació de Poincaré

Aquesta subsecció i les posteriors van dirigides principalment al càlcul d'òrbites periòdiques, elements importants dins l'*esquelet* del retrat de fases d'un sistema d'equacions diferencials. Seguirem essencialment el capítol 5 de [3], tot i que simplificat i adaptat al nostre cas. Començarem introduïnt l'anomenada *aplicació de Poincaré* que esdevindrà una eina bàsica d'aquest estudi.

Considereu l'equació diferencial $\dot{x} = f(x)$ amb $x \in \mathcal{U} \subset \mathbb{R}^n$, essent $\mathcal{U}$ un obert. Sigui $\phi(t, x)$ el seu flux associat, és a dir, verificant que

$$\frac{d}{dt}\phi(t,x) = f\left(\phi(t,x)\right), \qquad \phi(0,x) = x.$$

Sabem que si $f \in \mathcal{C}^r$ llavors $\phi \in \mathcal{C}^r$. Sigui Σ una hipersuperfície (secció) transversal al flux definida com $\Sigma = \{g(x) = 0\}$, amb g prou regular. El fet que sigui transversal a un punt x implica que el producte escalar $\langle \nabla g(x), f(x)\rangle \neq 0$ a tot punt de Σ. Definim aleshores la següent *aplicació de Poincaré*[1]: $P : \Sigma \to \Sigma$ a on $\bar{x} = P(x) = \phi(t(x), x) \in \Sigma$ i ho fa amb la mateixa orientació respecte de Σ de com ha sortit d'x (vegeu Fig 2). Recordeu que una manera de comprovar que l'orientació

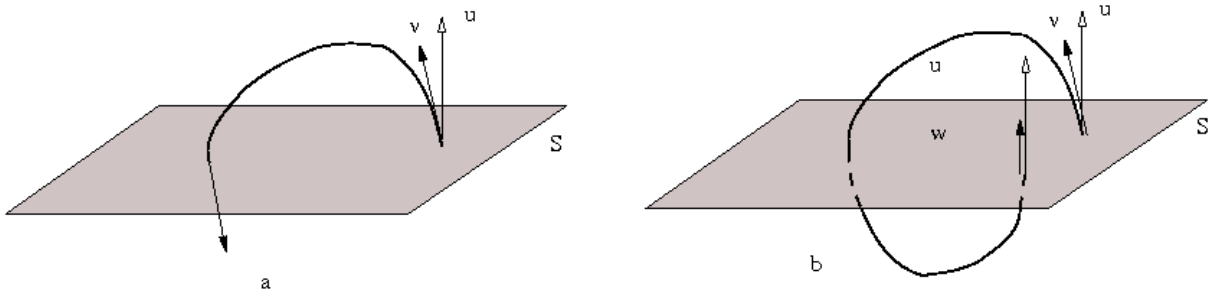

Figure 2: a) Intersecció del flux amb Σ amb orientació invertida; b) Idem però amb la mateixa orientació.

d'arribada és la mateixa que la de sortida és que

$$\langle \nabla g(x), f(x)\rangle \cdot \langle \nabla g(\bar{x}), f(\bar{x})\rangle > 0.$$

La hipersuperfície Σ s'anomena *secció de Poincaré*. Òbviament, el temps $t(x)$ necessari fins arribar a Σ (amb la mateixa orientació) depèn del valor inicial x i cal calcular-ho cada vegada. Noteu que $t(x)$ és solució de $\psi(t) = g(\phi(t, x)) = 0$. Per tant, per trobar-lo podem procedir de la següent manera:

[1] N'hi ha de diferents tipus, segons si ens interessa considerar el primer, segon, tercer, ... tall amb Σ o bé si el que volem és deixar actuar el flux $\phi(t, x)$ durant un cert temps fixat, etc

(*i*) Fent servir Taylor , calculem iterats del flux $x^{(1)}, x^{(2)}, \ldots$ on $x^{(j)} = \phi(t^{(j)}, x^{(0)})$ fins a arribar al primer iterat k que satisfà les següents dues condicions:

$$g(x^{(k)}) \cdot g(x^{(k-1)}) < 0, \qquad \langle \nabla g(x^{(k)}), f(x^{(k)}) \rangle \cdot \langle \nabla g(x^{(k-1)}), f(x^{(k-1)}) \rangle > 0.$$

La primera equival a dir que acabem de travessar Σ. La segona que ho hem fet amb la mateixa orientació que de sortida. Per tant, sabem que l'arrel $t(x)$ de $\psi(t) = g(\phi(t, x))$ que estem buscant es troba dins l'interval $\left[t^{(k-1)}, t^{(k)}\right]$.

(*ii*) Apliquem algun esquema iteratiu que ens permeti refinar aquesta arrel de ψ. Tenim diferents opcions: bisecció, secant, regula-falsi, Newton, ... Ho farem usant Newton doncs si l'interval inicial és prou petit la convergència serà quadràtica. Prenem com a punt de partida el valor $x^{(k-1)} = \phi(t^{(k-1)}, x)$ i tenint en compte que el flux compleix que $\phi(t + s, x) = \phi(s, \phi(t, x))$, l'esquema iteratiu corresponent és

$$s^{(m+1)} = s^{(m)} - \frac{\psi(s^{(m)})}{\psi'(s^{(m)})} = s^{(m)} - \frac{g(\phi(s^{(m)}, x^{(k-1)}))}{\langle \nabla g(\phi(s^{(m)}, x^{(k-1)}), f(\phi(s^{(m)}, x^{(k-1)}) \rangle}$$

amb condició inicial $s^{(0)} = t^{(k-1)}$ (vegeu Fig. 3). Un cop obtingut $s^* = \lim_m s^{(m)}$ tindrem que

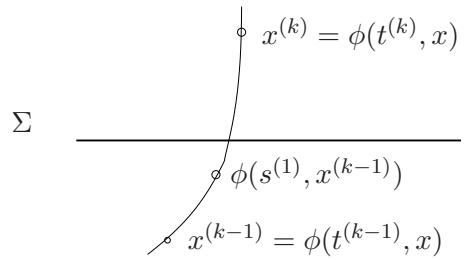

Figure 3: Refinant el punt de tall amb Σ

el temps buscat és $t(x) = t^{(k-1)} + s^*$. Finalment,

$$P(x) = \phi(t(x), x) = \phi(s^*, \phi(t^{(k-1)}, x)) = \phi(s^*, x^{(k-1)}). \tag{2}$$

Exercici 5.2 *Considereu la següent família de sistemes d'equacions diferencials*

$$\begin{cases} \dot{x} &= -f\left(\sqrt{x^2+y^2}\right) y \\ \dot{y} &= f\left(\sqrt{x^2+y^2}\right) x \end{cases} \tag{3}$$

on f és una funció $\mathcal{C}^\infty(\mathbb{R})$ tal que $f(0) = 0$.

1. *Demostreu que en coordenades polars l'equació (3) s'escriu*

$$\begin{cases} \dot{r} &= 0 \\ \dot{\theta} &= f(r) \end{cases}$$

L'origen és un punt d'equilibri (tipus centre) i totes les circumferències amb centre a l'origen són corbes invariants pel flux.

2. (*i*) *Considereu* $f(u) = 1$, $f(u) = u$ *i* $f(u) = u^2$. *Useu* Taylor *i* `gnuplot` *per a dibuixar aproximadament el seu retrat de fases entorn a l'origen: preneu condicions inicials sobre l'eix* $x = 0$ *i integreu l'edo per a temps positiu* τ *i per a temps negatiu* $-\tau$ ($\tau = 10$, *per exemple). Comproveu que surten circumferències concèntriques de centre* $(0, 0)$.

   (*ii*) *Considereu la secció de Poincaré* $\Sigma = \{y = 0, x > 0\}$. *Demostreu que el flux és sempre transversal a tot punt de* Σ. *Definiu* $x_k = 0.1(1 + k)$, $k = 0, 1, \ldots, 99$ *punts equiespaiats a l'interval* $[0, 10]$. *Per a cada* x_k *calculeu la seva imatge* $P(x_k)$ *per l'aplicació de Poincaré. Comproveu que* $P(x_k) = x_k$ *(això implica, en particular, que totes elles són òrbites periòdiques). Dibuixeu la funció* $t(x_k)$, *on* $t(x_k)$ *és tal que* $P(x_k) = \phi(t(x_k), x_k)$. *Què podeu deduir-ne?*

   (*iii*) *Feu el mateix prenent* $f(u) = u$ *i* $f(u) = u^2$. *Com són ara les gràfiques de* $t(x_k)$? *Compareu-les amb les de les funcions* $2\pi/u$ *i* $2\pi/u^2$, *respectivament.*

En aquest exemple, totes les òrbites són periòdiques (o.p.) i, concretament, el valor $t(x_k)$ coincideix amb el període de l'o.p. que passa per $(x_k, 0)$. La funció $T(x) = t(x)$ s'anomena *funció de període*. Es fàcil demostrar que $T(x) = 2\pi/f(x)$ (penseu-ho en polars). Al cas $f(u) = 1$ totes les o.p. tenen el mateix període, mentre que a la resta depén de la distància a l'origen. Al primer cas es diu que l'origen és un *centre isòcron* (del grec, *iso*=igual, *kronos*=temps). A la resta de casos es diu que tenim a l'origen un *centre anisòcron*.

5.3 Càlcul d'òrbites periòdiques. Zeros de l'aplicació de Poincaré

Considereu l'anomenat *oscil.lador de Van der Pol*,

$$\ddot{x} + \varepsilon(x^2 - 1)\dot{x} + x = 0 \tag{4}$$

amb $\varepsilon > 0$. Aquesta equació va ser introduïda per Van der Pol l'any 1927 per modelitzar el comportament de determinats circuits elèctrics i constitueix un exemple clàssic dins la Teoria de sistemes dinàmics al pla. Per estudiar-ho i com és habitual, considerem el sistema d'ordre 1 associat,

$$\begin{cases} \dot{x} &= y, \\ \dot{y} &= -x + \varepsilon(1 - x^2)y, \end{cases} \tag{5}$$

que té l'origen com a punt d'equilibri. L'anomenat sistema no pertorbat, $\varepsilon = 0$ correspon a un oscil.lador harmònic $\ddot{x} + x = 0$, les òrbites del qual són circumferències de centre $(0, 0)$ recorregudes amb freqüència angular 1. Es tracta doncs (vegeu Problema 5.2) d'un centre isòcron.

Per $\varepsilon \neq 0$ l'origen és també un punt d'equilibri, tot i que aquesta vegada inestable. És ben coneguda l'existència d'una única òrbita periòdica aïllada a l'entorn del $(0, 0)$. Aquest tipus d'o.p. s'anomenen *cicles-límit*. En el nostre cas, aquest cicle-límit és atractor, és a dir, totes les trajectòries amb valors inicials a prop d'ell tendeixen a ell quan $t \to +\infty$. A la Figura 4 es mostren algunes trajectòries (parcials, amb c.i. a sobre de la recta $\{y = 0\}$), tendint cap a aquest cicle-límit.

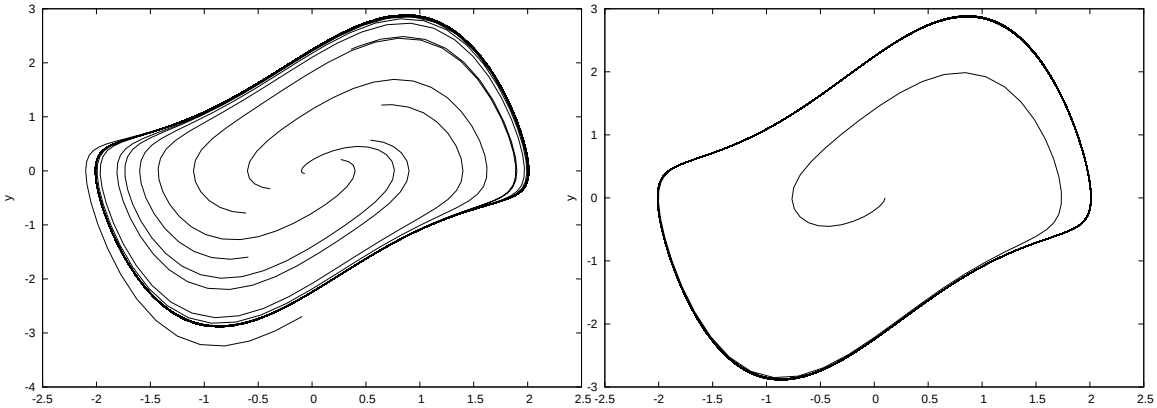

Figure 4: Algunes trajectòries de l'oscil.lador de Van der Pol $\ddot{x} + \varepsilon(x^2 - 1)\dot{x} + x = 0$ amb $\varepsilon = 1.2$

Exercici 5.3 *Considereu l'oscil.lador de Van der Pol, escrit en la forma (5).*

(*i*) *Preneu c.i. del tipus* $(x_k, 0)$, $x_k = -2.1 + k * 0.5$, $k = 0 \div 8$ *i trobeu el flux solució fins a temps* $t_f = 100$. *Feu el mateix integrant cap a darrere fins a* $t_f = -100$.

(*ii*) *Calculeu l'òrbita de condicions inicials* $(0.1, 0)$ *fins a temps* $t_f = 1000$. *Què s'observa?*

Indicacions:

1.- Escriviu un fitxer `vdpol.eq1` *amb les equacions del nostre sistema.*

2.- Demaneu a Taylor *que generi l'integrador i el header corresponents:*

```
perico@delospalotes: taylor -o vanderpol.c -jet -step -name vanderpol vdpol.eq1
perico@delospalotes: taylor -o taylor.h -header
```

3.- Feu els programes que es demanen als apartats (i),(ii) *anteriors, escrivint els resultats en dos fitxers anomenats* `retrat-fase-vdpol.dat` *i* `una-orbita-vdpol.dat`, *respectivament. Recordeu que, a*

banda de la compilació i linkat *des de línia de comandes, podeu fer-ho també a través d'un* `makefile`*. Per exemple, en el nostre cas, aquest fitxer, que anomenarem* `makefile-retrat-fases-vdpol`*, podria ser:*

```
CODIGO:= retrat-fases-vdpol.c vanderpol.c
CC:= gcc
CFLAGS:= -g -O3 -s
ENLACE:=$(CFLAGS) -lm
EJECUTABLE:=retrat-vdpol

OBJETOS:=$(CODIGO:.c=.o)
OBJETOS:=$(OBJETOS:.f=.o)

all:$(EJECUTABLE)

$(EJECUTABLE):$(OBJETOS)
$(CC) -o $(EJECUTABLE) $(OBJETOS) $(ENLACE)

clean:
rm -f $(OBJETOS) $(EJECUTABLE)
```

Per executar-ho feu

```
perico@delospalotes: make -f makefile-retrat-fases-vdpol
```

4.- Un cop teniu els fitxers .dat amb els punts de les trajectòries demanades, dibuixeu-les amb el `gnuplot`*. Podeu fer-ho directament a través de línia de comandes (vegeu Secció 5.1.1) o bé usar una* `shell`*: escriviu i deseu com a* `dibuixa-retrat-fases-vanderpol.sh` *el següent conjunt d'instruccions*

```
/usr/bin/gnuplot << EOF

set term postscript

set xlabel "x"
set ylabel "y"

set output "./retrat-de-fases-vanderpol.ps"
plot "./retrat-fase-vdpol.dat" u 2:3 notitle w l lw 0.1 lt 1

set output "./una-orbita-vdpol.ps"
```

```
plot "./una-orbita-vdpol.dat" u 2:3 notitle w l lw 0.1 lt 1

# Un exemple amb comandes no abreujades seria:
# plot "./retrat-fase-vdpol.dat" using 2:3 title 'Retrat de fases'
# with points point-type 1  point-size 0.4 line-type 1, 0 with lines
# line-width 0.1 line-type 1 notitle

# Recordeu que # indica comentari i la linia corresponent es obviada
# pel sistema.

EOF
```

Per executar-ho feu

```
perico@delospalotes: sh dibuixa-retrat-fases-vanderpol.sh.
```

Generarà dos fitxers `postscript` *que podreu visualitzar amb* `okular`*,* `ghostview`*,* `gv`*, etc.*

El nostre objectiu final en aquesta secció serà trobar numèricament el cicle-límit de l'oscil.lador de Van der Pol (que podeu intuïr a la Figura 4), esbrinar el seu periode i la seva estabilitat. Per fer-ho ens caldrà conèixer quina és la dependència de les solucions del nostre sistema (5) respecte a condicions inicials, determinada per l'anomenada *equació variacional*. Com ja hem comentat anteriorment seguirem [3].

5.3.1 Primera Equació Variacional d'una edo

Considereu una edo autònoma (és a dir, sense dependència explícita en el temps t al camp de vectors)

$$\dot{x} = f(x), \quad x \in \mathcal{U} = \mathring{\mathcal{U}} \subset \mathbb{R}^n, \ f \in \mathcal{C}^r(\mathcal{U}). \tag{6}$$

Si no diem res en contra, durant aquesta secció i les posteriors, suposarem que $\mathcal{U}$ i f són d'aquesta forma. Sigui $\phi(t,x)$ la solució de (6) satisfent que $\phi(0,x) = x$. Sabem que si $f \in \mathcal{C}^r$ aleshores $\phi \in \mathcal{C}^r$ i, per tant, té sentit fer la seva expansió en sèrie de Taylor entorn a x:

$$\phi(t, x+h) = \phi(t,x) + D_x\phi(t,x)h + \frac{1}{2}D_x^2\phi(t,x)h^2 + \cdots$$

S'anomenen *equacions variacionals* a aquelles equacions diferencials (lineals) satisfetes per les diferencials $D_x^j\phi(t,x)$. En particular, per a $j=1$ ens proporciona com varien les solucions de (6) en funció de les seves c.i. Una manera senzilla de deduir aquestes equacions és fer servir que $\phi(t,x)$ és solució, és a dir,

$$\frac{d}{dt}\phi(t,x) = f(\phi(t,x)).$$

Diferenciant aquesta expressió respecte de x,

$$D_x\left(\frac{d}{dt}\phi(t,x)\right) = Df(\phi(t,x)),$$

conmutant ambdues derivacions (respecte de t i de x, atès que $f \in \mathcal{C}^r$) i tenint en compte que $\phi(0,x) = x$, s'obté la *primera equació variacional*:

$$(\text{EV}) : \left\{ \begin{array}{rcl} \frac{d}{dt} D_x\phi(t,x) & = & Df(\phi(t,x)\, D_x\phi(t,x), \\ D_x\phi(0,x) & = & \text{Id}, \end{array} \right. \tag{7}$$

on Id denota la matriu identitat n-dimensional. De manera anàloga s'anirien trobant les equacions variacionals d'ordre superior. Des d'punt de vista computacional, és important destacar que s'integra numèricament (EV) juntament amb l'edo $\dot{x} = f(x)$, és a dir, en realitat s'integra el sistema ampliat

$$\frac{d}{dt}\begin{pmatrix} x \\ D_x\phi(t,x) \end{pmatrix} = \begin{pmatrix} f(x) \\ Df(\phi(t,x)D_x\phi(t,x) \end{pmatrix}$$

amb condicions inicials $\phi(0,x) = x$ i $Z(0) = \text{Id}$. Per exemple, per $n = 2$ el corresponent camp vectorial s'escriu

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2) \\ Df(\phi(t,x)Z \end{pmatrix} \tag{8}$$

amb

$$Z(t) = \begin{pmatrix} z_{11}(t) & z_{12}(t) \\ z_{21}(t) & z_{22}(t) \end{pmatrix}, \qquad Df(x_1,x_2) = \begin{pmatrix} D_1f_1(x_1,x_2) & D_2f_1(x_1,x_2) \\ D_1f_2(x_1,x_2) & D_2f_2(x_1,x_2) \end{pmatrix}.$$

Aleshores, a cada punt t_k cal integrar primer l'equació (8), trobar-ne el valor aproximat del flux $\phi(t_k,x) = (\phi_1(t_k,x), \phi_2(t_k,x))$ i amb ell obtenir-ne $Z(t_k)$ solució de $\dot{Z} = Df(\phi(t_k,x)Z$.

Exercici 5.4 *Trobeu el valor aproximat de $\phi(t_f,x)$ i de la seva primera variacional (EV) a temps $t_f = 10, 100, 1000$ i condicions inicials $x = (0.2, 0)$ i $x = (-1.5, 2)$.*

5.3.2 Càlcul de la diferencial de l'aplicació de Poincaré

Donada l'edo $\dot{x} = f(x)$ i una secció $\Sigma = \{g(x) = 0\}$ transversal al seu flux, considerem la seva aplicació de Poincaré associada (vegeu Secció 5.2):

$$\begin{array}{rccl} P: & \Sigma & \longrightarrow & \Sigma \\ & x & \longmapsto & P(x) := \phi(t(x),x) \end{array}$$

de manera que es verifica que $g(\phi(t(x),x) = 0$. La funció $t(x)$ és el temps necessari per a que l'òrbita amb c.i. x intersequi de nou i en el sentit establert (vegeu Figura 2) la hipersuperfície Σ. Volem

calcular la diferencial de l'aplicació de Poincaré P determinada per Σ. Diferenciant la seva definició obtenim

$$DP(x) = \frac{\partial}{\partial t}\phi(t(x),x)\ \frac{\partial t(x)}{\partial x} + D_x\phi(t(x),x) = f(\phi(t(x),x))\ \frac{\partial t(x)}{\partial x} + D_x\phi(t(x),x). \tag{9}$$

El terme $D_x\phi(t(x),x)$ és la solució de la (EV) per a temps $t(x)$ (vegeu Secció 5.3.1) i, per tant, computable. Necessitem doncs, calcular $\partial t(x)/\partial x$. Per fer-ho usem que $\phi(t(x),x) \in \Sigma$, és a dir, $g(\phi(t(x),x) = 0$. Diferenciant-la,

$$D_x\left(g\left(\phi(t(x),x)\right)\right) = 0 \Rightarrow D_x g\left(\phi(t(x),x)\right) \cdot \left(f(\phi(t(x),x))\ \frac{\partial t(x)}{\partial x} + D_x\phi(t(x),x)\right) = 0.$$

Si denotem $\bar{x} = \phi(t(x),x)$, aleshores, de l'anterior expressió s'obté

$$D_x g(\bar{x}) \cdot \left(f(\bar{x})\ \frac{\partial t(x)}{\partial x} + D_x\phi(t(x),x)\right) = 0 \Rightarrow$$

$$\langle \nabla g(\bar{x}), f(\bar{x})\rangle \frac{\partial t(x)}{\partial x} + Dg(\bar{x}) D_x\phi(t(x),x) = 0 \Rightarrow$$

$$\frac{\partial t(x)}{\partial x} = -\frac{1}{\langle \nabla g(\bar{x}), f(\bar{x})\rangle} Dg(\bar{x}) D_x\phi(t(x),x),$$

que està ben definida ja que $\langle \nabla g(\bar{x}), f(\bar{x})\rangle \neq 0$ en ser Σ, per hipòtesi, transversal al flux. Substituïnt aquesta expressió a l'equació (9) ens queda

$$DP(x) = \left(-\frac{1}{\langle \nabla g(\bar{x}), f(\bar{x})\rangle}\, f(\bar{x})\, Dg(\bar{x}) + \mathrm{Id}\right) D_x\phi(t(x),x), \tag{10}$$

essent Id la matriu identitat n-dimensional.

Anem a il.lustra-ho amb un exemple. Suposeu $x \in \mathbb{R}^n$ i que la nostra secció de Poincaré Σ, transversal al flux al domini considerat, vé definida $g(x) = x_n$, la n-èsima component d'x. És a dir, $\Sigma = \{x \in \mathbb{R}^n \mid x_n = 0\}$. Aleshores, a (10) simplificant, tindrem que $\nabla g(x) = (0,0,\ldots,1) = e_n$, $\langle \nabla g(\bar{x}), f(\bar{x})\rangle = f_n(\bar{x})$,a on $f = (f_1, f_2, \ldots, f_n)$, i $Dg(\bar{x}) = (0\ 0\ 0 \ldots 1)$. A més,

$$f(\bar{x}) \cdot Dg(\bar{x}) = \begin{pmatrix} 0 & 0 & \ldots & 0 & f_1(\bar{x}) \\ 0 & 0 & \ldots & 0 & f_2(\bar{x}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & f_n(\bar{x}) \end{pmatrix}$$

Per tant,

$$DP(x) = \begin{pmatrix} 1 & 0 & \ldots & 0 & -f_1(\bar{x})/f_n(\bar{x}) \\ 0 & 1 & \ldots & 0 & -f_2(\bar{x})/f_n(\bar{x}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & -f_{n-1}(\bar{x})/f_n(\bar{x}) \\ 0 & 0 & \ldots & 0 & 0 \end{pmatrix} D_x\phi(t(x),x) =: A \cdot Z,$$

a on $Z = D_x\phi(t(x), x)$ és la solució de l'equació variacional (EV) (vegeu (7)) i $A = (a_{ij})_{1\leq i,j\leq n}$ té coeficients

$$\begin{cases} a_{ii} &= 1 \\ a_{in} &= -\dfrac{f_i(\bar{x})}{f_n(\bar{x})} \end{cases} \qquad \text{per } i = 1, 2, \ldots, n-1,$$

i $a_{ij} = 0$ a la resta de casos. Si denotem $Z = (z_{ij})_{1\leq i,j\leq n}$ i $DP(x) = (q_{ij})_{1\leq i,j\leq n}$ aleshores els coeficients de la diferencial de Poincaré s'obtenen fent $q_{ij} = \sum_{k=1}^{n} a_{ik} z_{kj}$.

5.3.3 Càlcul d'òrbites periòdiques

Es vol calcular numèricament òrbites periòdiques d'un cert periode T d'un sistema (autònom) $\dot{x} = f(x)$, $x \in \mathbb{R}^n$. La primera idea seria la de cercar condicions inicials x_0 que siguin zero de la funció $G(x_0) = \phi(T, x_0) - x_0$ i aplicar, per fer-ho, el mètode de Newton (falta referència). Malauradament (o no), si x_0 satisfà que $\phi(T, x_0) = x_0$ llavors també ho verifica tot punt de la seva òrbita $\{\phi(t, x_0)\}_{t\in\mathbb{R}}$. Això provoca, en particular, que el vector tangent a l'òrbita al punt x_0, $f(x_0)$, sigui vector propi de valor propi 1 de la *matriu de monodromia* $D\phi(T, x_0)$. Conseqüentment, la matriu $DG(x_0) = D\phi(T, x_0) - \mathrm{Id}$ és singular. Per tant no podem aplicar directament el mètode de Newton i caldria buscar alguna de les seves variants. No serà aquesta l'opció que aplicarem en aquesta secció sinó que buscarem treure profit de l'aplicació de Poincaré definida anteriorment.

Sigui de nou Σ una secció transversal al flux de $\dot{x} = f(x)$. Observeu que buscar una òrbita periòdica (que intersequi Σ, és clar) és equivalent a cercar un punt $x_0 \in \Sigma$ de forma que $P(x_0) = x_0$, on P és l'aplicació de Poincaré associada a $\dot{x} = f(x)$ i a Σ. Noteu, a més, que el periode de l'òrbita periòdica que estem buscant no està ara prefixat. Si $P(x_0) = x_0$ tindrem que $T = t(x_0)$, essent $t(x_0) >$ el temps que fa $\phi(t(x_0), x_0) \in \Sigma$ amb l'orientació fixada. Per tant, aplicarem el mètode de Newton a la funció $G(x_0) = P(x_0) - x_0$, que té per diferencial $DG(x_0) = DP(x_0) - \mathrm{Id}$. Recordeu que a la Secció 5.3.2 ja vàrem explicar com calcular-la.

Cal tenir present que, de vegades, la diferencial $DP(x_0)$ de l'aplicació de Poincaré té 1 com a valor propi i, llavors, $DG(x_0)$ és singular. Aquest fet és sovint motivat per l'existència d'*integrals primeres*[2] del nostre sistema $\dot{x} = f(x)$. Un exemple clàssic d'integral primera és l'energia E (potencial + cinètica) d'un sistema mecànic no dissipatiu. En aquests casos caldrà restringir l'estudi considerant punts d'un nivell d'energia (o de la integral primera) fixat, $E = E_0$.

L'ús de l'aplicació de Poincaré P en el càlcul d'òrbites periòdiques de $\dot{x} = f(x)$ esdevé també de gran utilitat per a determinar la seva estabilitat lineal. De fet, l'estabilitat de l'òrbita periòdica associada al punt $x_0 \in \Sigma$ és equivalent a l'estabilitat lineal de x_0 com a punt fix de l'aplicació P. Així, si tot *vap* de $DP(x_0)$ verifica que $|\lambda| < 1$ o si $|\lambda| = 1$ que sigui simple, aleshores x_0 és *estable*

[2]Recordeu que una funció escalar F és una integral primera de $\dot{x} = f(x)$ si és constant sobre cada una de les seves solucions: $\dfrac{d}{dt}\, F(\phi(t, x)) = 0$.

i, per tant, també la seva òrbita (periòdica). D'igual manera, si algun *vap* λ té mòdul més gran que 1 llavors ambdós són *inestables*.

Exercici 5.5 *Preneu* $\Sigma = \{y = 0\}$ *com a secció de Poincaré per a l'oscil.lador de Van der Pol* (4) *i trobeu l'única òrbita periòdica (aïllada, és a dir, un cicle-límit) per a diversos valors del paràmetre* ε *(per exemple:* $1.2, 0.5, 0.25, 0.01$*). Estudieu la seva estabilitat (lineal).*

Bibliografia

[1] À. Jorba, M. Zou A software package for the numerical integration of ODE by means of high-order Taylor methods, (pdf)

[2] À. Jorba, M. Zou Taylor User's Manual, (pdf)

[3] C. Simó. On the analytical and numerical approximation of invariant manifolds. Reprinted from *Les Mêthodes Modernes de la Mecánique Céleste* (course given at Goutelas, France, 1989), D. Benest and C. Froeschlé (eds.), 285–329, Editions Frontières, Paris, 1990. (gzip)