

Càlcul Numèric i Simulació

Antoni Susín

Índex

Introducció	5
1 Errors	7
1.1 Error de representació. Error de mesura	7
1.1.1 Tipus de variables en C	8
1.2 Punt flotant	8
1.3 Fórmula de propagació de l'error	9
2 Iteració	11
2.1 Iteració d'una funció. Representació gràfica	12
2.2 Representacions	12
2.3 Punts fixos	13
2.4 Punts periòdics	14
2.5 El model logístic de creixement d'una espècie	15
2.5.1 Punts fixos	16
2.5.2 Estudi del comportament en funció de λ	16
3 Aplicacions de la Iteració	19
3.1 Càlcul de zeros de funcions	19
3.1.1 Mètode de bisecció	19
3.1.2 Mètodes de punt fixe	19
3.1.3 Acceleració de la convergència	21
3.1.4 Mètode de Newton	22
3.1.5 Mètode de la secant (regular falsi)	23
3.2 Sistemes lineals	23
Mètode de Jacobi	24
Mètode de Gauss-Seidel	24
Convergència dels mètodes	25
3.2.1 Mètode de Newton per a sistemes d'equacions (no lineals)	25
3.3 Generació de nombres aleatoris	26
3.3.1 Generació de lleis uniformes $U([0, 1])$	27

	Mètode de la barreja	28
3.3.2	Generació de lleis no uniformes	28
	Llei exponencial	28
	Llei normal	29
4	Equacions diferencials ordinàries. Integració numèrica	31
4.1	E.d.o. de primer ordre	31
4.2	Camp vectorial. Isoclines	31
4.3	Càlcul de solucions	32
	Mètode d'Euler	32
	Mètode d'Euler modificat	33
	Mètode d'Euler enrera	34
	Mètodes de Runge-Kutta	35
	Anàlisi de l'error	35
	Mètodes de Runge-Kutta	37
4.3.1	Integració automàtica. Control de pas	37
	Mètode de Runge-Kutta-Fehlberg (RK45)	37
	Utilització de l'integrador RK45F	39
4.3.2	Retrat de Fase d'un sistema d'equacions diferencials ordinàries. Ob- jectes invariants	41
	Objectes invariants	41
	Punts d'equilibri	41
	Òrbites periòdiques	41
	Objectes invariants en $\dim \geq 3$	41
4.3.3	Càlcul numèric d'òrbites periòdiques. Secció de Poincaré	41
4.3.4	Models d'interacció	42
	Model de Malthus	42
	Model logístic	43
	Estudi qualitatiu d'un model depredador-presa	44
	Punts d'equilibri	44
	Isoclines (de velocitat zero)	44
	Caràcter dels punts d'equilibri	45
5	Apendix A: Implementació en llenguatge C del mètode RK45F	47

Introducció

El curs de càlcul numèric i simulació pretén introduir a l'alumne en la vessant de les matemàtiques que té una estreta relació amb els ordinadors i la programació. De fet al llarg del curs veurem diferents temes com la resolució d'equacions, càlcul dels zeros d'un polinomi, sistemes d'equacions lineals, generació de nombres aleatoris, etc.

Volem fer èmfasi en els càlculs en els que la solució analítica o bé no existeix, o bé no és fàcil calcular-la.

Un capítol destacat estarà dedicat a la resolució numèrica d'equacions diferencials ordinàries. Estudiarem els diferents mètodes que existeixen i utilitzarem principalment el RKF45 (Runge-Kutta-Felberg d'ordre 4 i 5) amb control automàtic de pas.

Al llarg de tot el curs donarem una gran importància a la representació gràfica dels resultats, per la qual cosa, utilitzarem una llibreria gràfica (OPENGL) que complementarà els aspectes relacionats amb la programació. El llenguatge **C** és el que farem servir per elaborar les pràctiques de l'assignatura. Aprofundirem el coneixement que els alumnes tenen d'aquest llenguatge i farem èmfasi en els aspectes numèrics del mateix.

Eines:

- Llenguatge de programació → **C**
- Compilador → **Visual C++**, (windows 95/98/XP))
- Llibreria gràfica → **OpenGL**
- Gestor de finestres i animació → **Glut**

Capítol 1

Errors

Definició 1.0.1 Anomenarem error absolut d'una certa magnitud \bar{x} a la diferència $e_a(x) = |\bar{x} - x|$, on \bar{x} és el veritable valor i x és el valor mesurat o calculat.

Definició 1.0.2 Anomenarem error relatiu a $e_r(x) = \frac{e_a(x)}{|\bar{x}|} \left(\simeq \frac{e_a(x)}{|x|} \right)$.

1.1 Error de representació. Error de mesura

Des de'l punt de vista numèric l'error comès està molt lligat amb la *representació* que es fa dels nombres dins un ordinador:

$$\frac{1}{10} = 0.1, \quad \frac{1}{3} = 0.33\dots 3, \quad \frac{1}{6} = 0.1666\dots 67$$

$10_{10} \longrightarrow 2^3 + 2 = 1\ 0\ 1\ 0_2$	$1 \longrightarrow 1_2 \longrightarrow 2^0$
	$2 \longrightarrow 10_2 \longrightarrow 2^1$
	$3 \longrightarrow 11_2 \longrightarrow 2^1 + 2^0$
	$4 \longrightarrow 100_2 \longrightarrow 2^2$
	$5 \longrightarrow 101_2 \longrightarrow 2^2 + 2^0$

Passar a base 2:

$$65_{10} \longrightarrow \begin{array}{r} 65 \\ 05 \\ 1) 12 \\ \quad 0) \\ \quad \quad 0) \\ \quad \quad \quad 0) \\ \quad \quad \quad \quad 0) \\ \quad \quad \quad \quad \quad 0) \\ \quad \quad \quad \quad \quad \quad 0) \\ \quad \quad \quad \quad \quad \quad \quad 1 \end{array} \qquad 1000001_2 = 2^6 + 2^0$$

$$\begin{aligned}
0.1_{10} &\longrightarrow 0.1 \times 2 = 0.2 \Rightarrow 0 \\
&\quad 0.2 \times 2 = 0.4 \Rightarrow 0 \\
&\quad 0.4 \times 2 = 0.8 \Rightarrow 0 \\
&\quad 0.8 \times 2 = 1.6 \Rightarrow 1 \\
&\quad 0.6 \times 2 = 1.2 \Rightarrow 1 \\
&\quad 0.2 \times 2 = 0.4 \Rightarrow 0 \\
&\implies 0.1_{10} = 0.000110011_2 \dots
\end{aligned}$$

- **Longitud d'una paraula:** 32 bits \equiv 4 bytes (8 bits=1 byte).

1.1.1 Tipus de variables en C

Tipus	Memòria (bits)	rang
char	8 (1 byte)	$-128 \div 127$
int	16	$-32768 \div 32767$
unsigned int	16	$0 \div 65535$
long int	32	$-2147483648 \div 2147483647$
unsigned long int	32	$0 \div 4294967295$
float	32	$3.4e - 38 \div 3.4e + 38$ (i els negatius)
double	64	$1.7e - 308 \div 1.7e + 308$ (i els negatius)

Observació 1.1.1 Si `int i;`

$$\begin{aligned}
i &= 32767; \\
i &= i + 1; \quad i? \quad (\implies i = -32768 !!!)
\end{aligned}$$

1.2 Punt flotant

Un nombre real el podem representar de la forma:

- per exemple: $x = 0.01234 \iff \text{fl}(x) = 1.234 \cdot 10^{-2}$
- en general: $\underbrace{d_0 \cdot d_1 d_2 d_3 d_4 \dots d_{n-1}}_{\text{mantissa}} \text{exp}$ en funció de les xifres que tingui la mantissa.

Exemple 1.2.1 Si utilitzem 4 xifres de mantissa (*aritmètica* de 4 xifres):

- (a) Representem els valors $A = 123.4567$, $B = 0,01265$

$$\text{fl}(A) = 1.235 \cdot 10^{+2} \quad \text{fl}(B) = 1.265 \cdot 10^{-2}.$$

- (b) Calculem l'error absolut i relatiu de la suma $A + B$:

$$\begin{aligned}
A + B &= 123.46935 \implies \text{fl}(A + B) = 1.235 \cdot 10^2 \equiv \text{fl}(A) \\
e_a(A + B) &= (\text{fl}(A + B) - (A + B)) = 0.03065 \\
e_r(A + B) &= \frac{0.03065}{\text{fl}(A + B)} = 2.4817 \cdot 10^{-4}
\end{aligned}$$

La representació dins l'ordinador es fa repartint els 32 bits

1	8	23
0	01010101	001...110000
signe	exp	mantissa

float x ; 32 bit (8 xifres decimals)

1	8	55

double x ; 64 bit (16 xifres decimals)

1.3 Fórmula de propagació de l'error

Si tenim un valor x que coneixem amb un cert *error absolut* e_a i a partir d'ell en calculem un de nou $y = f(x)$, aleshores l'error absolut comès en calcular y ens el dona la *Fórmula de propagació de l'error*

$$|e_a(y)| = |f'(x)| \cdot e_a(x).$$

En general, si x_1, \dots, x_n són dades amb error absoluts $e_a(x_1), \dots, e_a(x_n)$, llavors l'error absolut comès en calcular $y = F(x_1, x_2, \dots, x_n)$ serà:

$$|e_a(y)| = \sum_{i=1}^n \left| \frac{\partial F}{\partial x_i}(x_1, \dots, x_n) \right| e_a(x_i).$$

Aplicació 1.3.1

1. $e_a(x_1 + x_2) = e_a(x_1) + e_a(x_2)$.

Demostració: $y = F(x_1, x_2) = x_1 + x_2 \quad \frac{\partial F}{\partial x_1} = 1 \quad \frac{\partial F}{\partial x_2} = 1$

$$e_a(y) = \frac{\partial F}{\partial x_1} e_a(x_1) + \frac{\partial F}{\partial x_2} e_a(x_2) = e_a(x_1) + e_a(x_2).$$

2. $e_a(\ln(x)) = e_r(x)$.

Demostració: $e_a(y) = \frac{1}{|x|} e_a(x) = e_r(x)$

3. $e_r(x_1 x_2) = e_r(x_1) + e_r(x_2)$.

Demostració: $e_r(x_1 x_2) = e_a(\ln(x_1 x_2)) = e_a(\ln(x_1)) + e_a(\ln(x_2)) = e_r(x_1) + e_r(x_2)$

Exercicis 1.3.2

1. Calculeu les expressions per l'error absolut corresponents al producte i la divisió:

$$e_a(x_1 x_2), \quad e_a(x_1/x_2)$$

2. Anàlogament per l'error relatiu de la divisió:

$$e_r(x_1/x_2)$$

3. Donada la funció $y = e^{3x}$, calculeu l'error comès en calcular y en el punt $x = 0.1 \pm \underbrace{0.001}_{e_a(x)}$.

4. Si $a = 1.0 \pm 0.01$ i $b = 3.0 \pm 0.02$, calculeu l'error absolut de l'expressió $y = \frac{4a + 5b}{a^2 + b^2}$.

5. Doneu una fita superior de l'error de les solucions del sistema $\begin{cases} 3x + ay = 1 \\ bx + y = 2 \end{cases}$ si

$$a = 1.0 \pm 10^{-3}$$
$$b = 2.0 \pm 10^{-2}$$

Capítol 2

Iteració

Definició 2.0.3 Anomenarem procés iteratiu (o recursiu) a una successió de valors que segueix un esquema de la forma $x_{n+1} = f(x_n)$, de manera que a partir d'un valor inicial (o llavor) x_0 es calculen la resta de valors

$$x_1 = f(x_0), \quad x_2 = f(x_1), \quad x_3 = f(x_2), \quad \dots$$

A la successió de valors $(x_n)_n$ obtinguts a partir d'un valor inicial x_0 se l'anomena òrbita del punt x_0 .

Els processos iteratius apareixen en molts contextes diferents

1. Solucions de sistemes d'equacions lineals. (Mètodes de Jacobi i Gauss-Seidel).
2. Càlcul de zeros de funcions.
3. Solucions numèriques d'equacions diferencials.
4. Càlcul de valors i vectors propis.

Normalment les preguntes que ens farem respecte a qualsevol provés iteratiu són les següents:

- Què passa quan es fan iteracions llargues?
- Com depèn de la llavor el què passa?
- Com depèn de la funció $f(x)$ el què passa?
- Com depèn dels possibles paràmetres dins de $f(x)$ el què passa?

Aquestes qüestions són similars a les que podríem fer-nos sobre les solucions (òrbites) d'una equació diferencial. De fet podem establir una relació entre equacions diferencials i iteracions.

Direm que les equacions diferencials descriuen l'evolució per un *temps continu* i les iteracions per un *temps discret* (Sistemes Dinàmics).

2.1 Iteració d'una funció. Representació gràfica

Tot procés iteratiu $x_{n+1} = f(x_n)$ queda caracteritzat per la funció $f(x)$ que el defineix. D'aquesta manera es pot parlar de la iteració d'una funció utilitzant l'equivalència

$$\begin{aligned} x_{n+1} &= f(x_n) = f(f(x_{n-1})) = f(f(f(x_{n-2}))) = \dots = (f \dots) f(x_0) = \\ &= (f \circ f \circ \dots \circ f)(x_0) = f^n(x_0) \end{aligned}$$

Observacions 2.1.1

§

1. Calcular el n -èssim iterat de x_0 és el mateix que aplicar f^n directament a x_0 .
2. $f^n(x_0) \neq (f(x_0))^n$. No confondre amb la potència.

Exemples 2.1.2

1. $x_{n+1} = \alpha x_n \implies f(x) = \alpha x$

$$f^2(x) = (f \circ f)(x) = \alpha(\alpha x) = \alpha^2 x$$

$$f^3(x) = (f \circ f \circ f)(x) = \alpha(\alpha(\alpha x)) = \alpha^3 x$$

...

$$f^n(x) = \alpha^n x$$

2. $x_{n+1} = x_n^2 + c \implies f(x) = x^2 + c$

$$f^2(x) = (x^2 + c)^2 + c$$

$$f^3(x) = ((x^2 + c)^2 + c)^2 + c$$

...

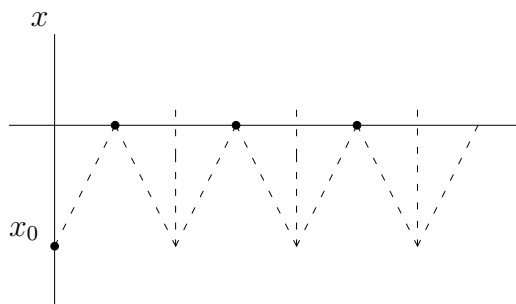
$$f^n(x) = ???$$

2.2 Representacions

Hi ha diferents formes de representar un procés iteratiu; introduïrem dos tipus:

- (I) *Sèries temporals*. Es tracta, simplement, de representar els punts $(n, x_n) \equiv (n, f^n(x_0))$, usualment unint-los per línies

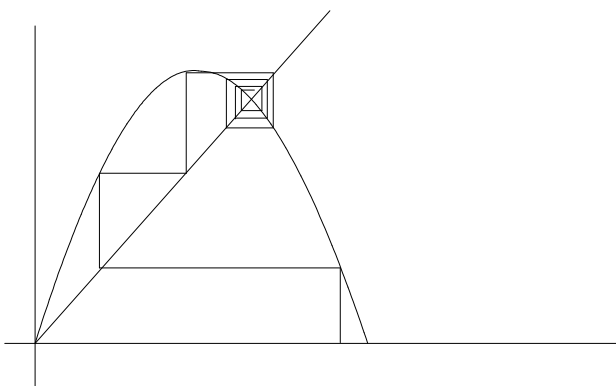
Exemple 2.2.1 $f(x) = x^2 - 1$, $x_0 = -1$, $x_1 = (-1)^2 - 1 = 0$, $x_2 = 0^2 - 1 = -1$, $x_3 = 0$, ...



Exercici 2.2.2 Fer-ho per a $x^2 - 1.3$.

(II) *Iteració gràfica.* Es pot representar gràficament una iteració passant successions de la gràfica de $f(x)$ a la recta $y = x$

- (i) Anar verticalment a la gràfica de $f(x)$, $(x_0, f(x_0)) = (x_0, x_1)$.
- (ii) Anar horitzontalment a la gràfica de $y = x$, $(f(x_0), f(x_0)) \equiv (x_1, x_1)$.



2.3 Punts fixos

Definició 2.3.1 Anomenarem punts fixos o bé punts d'equilibri aquells que satisfan $x_f = f(x_f)$.

Observació 2.3.2 Els punts fixos són les interseccions de la gràfica de $f(x)$ amb la recta $y = x$.

Definició 2.3.3 Direm que un punt fix x_f és atractiu (o estable) si prenent x_0 en un entorn de x_f , les seves òrbites tendeixen cap a x_f .

Direm que x_f és un repulsor si prenent x_0 en un entorn de x_f , les seves òrbites surten de l'entorn.

Proposició 2.3.4 Si diem $m = f'(x_f)$ on x_f és un punt fix, llavors si:

- $|m| < 1 \implies x_f$ és atractiu
- $|m| > 1 \implies x_f$ és repulsor
- $|m| = 1 \implies$ indiferent (possibles valors de bifurcació)
- $m = 0 \implies$ superatractor

Exemple 2.3.5

$$1. f(x) = x^2 - 1.2 \implies \text{punts fixes } p_1 = 1.7041\dots, p_2 = -0.7041\dots$$

$$\begin{aligned} f'(p_1) &= 3.4083 \\ f'(p_2) &= -1.4083 \implies p_1, p_2 \text{ repulsors.} \end{aligned}$$

$$2. f(x) = x^2 + c \equiv x$$

$$x^2 - x + c = 0, \quad x_f = \frac{1 \pm \sqrt{1 - 4c}}{2} = \frac{1 \pm \sqrt{3/2}}{2} \quad (\text{si } c = \frac{1}{8})$$

$$f'(x_f) = 2x_f = 1 \pm \sqrt{1 - 4c} = 1 \pm \sqrt{3/2}.$$

2.4 Punts periòdics

Definició 2.4.1 Direm que l'òrbita d'un punt x_0 per una iteració $f(x)$ és periòdica (o forma un cicle) de període n si, i només si,

$$x_{i+1} = f(x_i), \quad i = 0 \div n - 2 \quad \text{i} \quad f(x_{n-1}) = x_0$$

Observació 2.4.2 Això és equivalent a dir que tots els punts d'una òrbita periòdica són punts fixos de la funció $f^n(x)$ i no són fixos per cap altre iterat de f menor que n .

Definició 2.4.3 Direm que una òrbita és atractora (o repulsora) si ho són els seus punts per f^n .

Exemples 2.4.4

$$1. f(x) = x^2 - \frac{7}{4}$$

$$x_0 = \frac{1}{2} \implies x_1 = -\frac{3}{2} \implies x_2 = \frac{1}{2} \quad \text{òrbita 2-periòdica}$$

$$2. f(x) = x^2 - 1.75488\dots$$

$$x_0 = 0 \implies x_1 = -1.75488\dots \implies x_2 = 1.3247\dots \implies x_3 = 1.75488\dots$$

Proposició 2.4.5 Una determinada òrbita periòdica és atractora (o repulsora) si, i només si, el multiplicador $m_n = \prod_{i=0}^{n-1} f'(x_i) = f'(x_{n-1})f'(x_{n-2})\cdots f'(x_0)$ satisfà $|m_n| < 1$ (> 1 respectivament).

Demostració:

$$(f^n)'(x_0) \stackrel{\text{regla cadena}}{=} f'(f(\cdots(f(x_0)))) = f'(f(\cdots f(x_0)))(f \cdots f)'(x_0) = \cdots = f'(x_{n-1}) \cdots f'(x_0).$$

Exemple 2.4.6 $f(x) = x^2 - \frac{7}{4} \implies f'(x) = 2x$

$$x_0 = \frac{1}{2} \implies x_1 = -\frac{3}{2} \implies m_2 = (2x_0)(2x_1) = 4x_0x_1 = -3 \implies \text{cicle repulsor}$$

Exercici 2.4.7 Volem caracteritzar els punts x_p de període 2 per la iteració general $f(x) = x^2 + c$

1. Doneu els valors de c pels quals tenim valors x_p reals ($c < -3/4$).
2. Calculeu el multiplicador m_2 .
3. Digueu per a quins valors de c els punts periòdics són atractors

2.5 El model logístic de creixement d'una espècie

Suposem que tenim una espècie, la població de la qual s'avalua de generació en generació. Denotarem per p_n la població de l' n -èsima generació.

Considerem que la població següent té un augment proporcional a la població de la generació anterior:

$$p_{n+1} = p_n + \alpha p_n = (1 + \alpha)p_n.$$

Aquesta primera aproximació del model ens dona un *creixement* del tipus $p_{n+1} = (1 + \alpha)^n p_0$ que, si $\alpha > 0$, és *exponencial*.

Per millorar el model, considerem que hi ha un valor de saturació p_s que el medi pot tolerar, llavors podríem escriure

$$p_{n+1} = \left(1 + \alpha \frac{p_s - p_n}{p_s}\right) p_n$$

que, expressant-ho en funció de $\bar{p}_n = \frac{p_n}{p_s}$, ens quedaria

$$\bar{p}_{n+1} = (1 + \alpha)\bar{p}_n - \alpha\bar{p}_n^2.$$

Finalment, si volem fer el canvi $\bar{p} = \frac{1 + \alpha}{\alpha}x$, obtenim

$$\frac{1 + \alpha}{\alpha}x_{n+1} = (1 + \alpha) \left(\frac{1 + \alpha}{\alpha}x_n\right) - \alpha \left(\frac{1 + \alpha}{\alpha}x_n\right)^2 = (1 + \alpha)(x_n - x_n^2) \underset{1 + \alpha = \lambda}{=} \lambda x_n(1 - x_n)$$

$$\boxed{f(\lambda, x) = \lambda x(1 - x)}$$

Observació 2.5.1 $f(\lambda, 0) = f(\lambda, 1) = 0, \quad \forall \lambda$

$$f'(\lambda, x) = \lambda - 2\lambda x.$$

2.5.1 Punts fixos

Els únics punts fixos són $\bar{x} = 0$ i $\bar{x}_\lambda = 1 - \frac{1}{\lambda}$; si considerem que $\lambda > 1$ aquests punts estaran dins l'interval $(0, 1)$.

Per aquests valors, tindrem $f'(\lambda, 0) = \lambda$ i $f'(\lambda, \bar{x}_\lambda) = 2 - \lambda$.

Proposició 2.5.2

1. Per a $\lambda > 1$ tindrem que si $x_0 \notin (0, 1)$ llavors $f^n(\lambda, x_0) \xrightarrow[n \rightarrow \infty]{} -\infty$.

Demostració. Si $x_0 < 0$, aleshores $f(\lambda, x_0) < x_0 \rightarrow f^n(\lambda, x_0)$ és una successió decreixent que no pot convergir ja que no hi ha punts fixes negatius.

Si $x_0 > 1$, aleshores $f(\lambda, x_0) < 0$ i s'aplicarà el cas anterior.

2. Per a $1 < \lambda < 4$ tindrem que si $x_0 \in (0, 1)$, llavors $f^n(\lambda, x_0) \in (0, 1)$, per a tot n .
Si calculem el màxim de $f(x)$, llavors $f'(x) = \lambda(1 - 2x) \equiv 0 \implies x_m = 1/2$

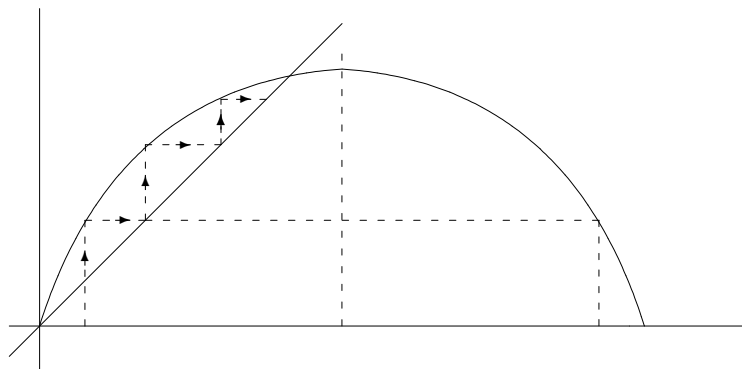
$$f(1/2) = \frac{\lambda}{2} \left(1 - \frac{\lambda}{2}\right) = \frac{\lambda}{4} \underset{1 < \lambda < 4}{<} 1.$$

2.5.2 Estudi del comportament en funció de λ

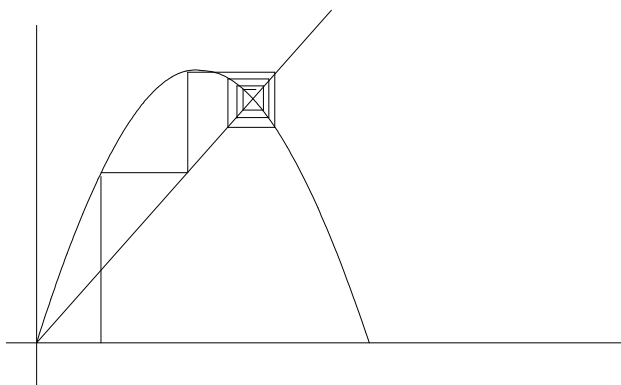
◇ $1 < \lambda < 3$

Els punts crítics són $x = 0$, repulsor i $\bar{x}_\lambda = 1 - \frac{1}{\lambda}$, atractor.

Observació 2.5.3 Per a $1 < \lambda < 2$ el màxim està per sota la bisectriu. Si $0 < x_0 \leq 1/2$ tendeix de forma creixent



Per a $2 < \lambda < 3$ el màxim està per sobre la diagonal. Convergència no monòtona.



◇ $\lambda = 3$

El punt fix $x = 0$ és repulsor, però no podem decidir sobre l'estabilitat de \bar{x}_λ ja que $f'(\bar{x}_\lambda) = -1$. (Fent una simulació es veu que segueix essent estable).

El valor $\lambda = 3$ s'anomena valor de *bifurcació*, ja que l'estabilitat canvia.

◇ $3 < \lambda < 1 + \sqrt{6} \simeq 3.449$

Per a aquests valors $x = 0$ i \bar{x}_λ són *inestables*; això vol dir que $f^n(x_0)$ no tendirà a *cap punt fix*.

Per a $\lambda = 3$ es produeix el que s'anomena una *bifurcació de període doble*, és a dir, el punt fix \bar{x}_λ dona lloc a una *òrbita periòdica* de període 2. Per estudiar l'estabilitat d'aquesta òrbita periòdica considerem els punts fixos de f^2 :

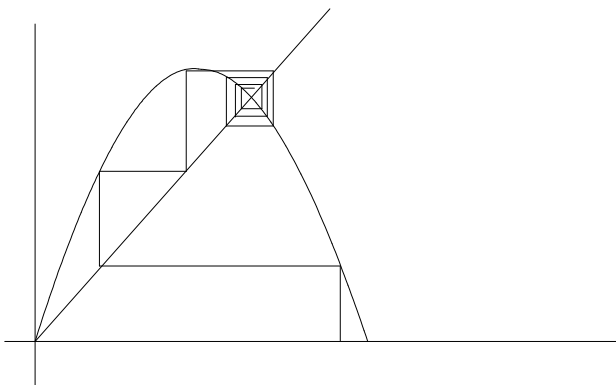
$$f^2(\lambda, x) - x = \lambda(\lambda x(1-x)) - (1 - \lambda x(1-x)) - x = 0$$

Com $x = 0$ és solució. $\implies \lambda^2(1-x)(1-\lambda x(1-x)) - 1 = 0$

Com $\bar{x}_\lambda = 1 - \frac{1}{\lambda} = 0$ és solució $\implies \lambda^2 x^2 - \lambda(\lambda+1)x + \lambda+1 = 0$

$$x_2 = \frac{\lambda(\lambda+1) \pm \sqrt{\lambda^2(\lambda+1)^2 - 4\lambda^2(\lambda+1)}}{2\lambda^2} = \frac{\lambda+1 \pm \sqrt{(\lambda+1)(\lambda+3)}}{2\lambda} - \dots$$

Es pot veure que per a $3 < \lambda < 1 + \sqrt{6}$ l'òrbita periòdica és estable



◇ $3.449 \simeq 1 + \sqrt{6} < \lambda < 3.570$

Per als valors $1 + \sqrt{6}$ l'òrbita 2-periòdica anterior perd la seva estabilitat i a la vegada es bifurca donant lloc a una òrbita periòdica estable de *període 4*, de fet per als valors $\lambda_1 = 3$,

$\lambda_2 = 3.499$, $\lambda_3 = 3.544$, $\lambda_4 = 3.564$, per a cada λ_k , tenim òrbites periòdiques de període 2 (és el que s'anomena *cascada de bifurcacions*). Es pot veure que $\lambda_k \xrightarrow{k \rightarrow \infty} \lambda_\infty \simeq 3.570$ i se satisfà la *constant de Feigenbaum*:

$$\lim_{k \rightarrow \infty} \frac{\lambda_k - \lambda_{k-1}}{\lambda_{k+1} - \lambda_k} = 4.6692\dots$$

◇ $\lambda > 3.570$

Regió caòtica: no es pot dir res (cascada de Feigenbaum).

Observació 2.5.4 Per a $\lambda = 3.839$, òrbita estable 3-periòdica

$$x_0 = 0.149888 \longrightarrow x_1 = 0.489172 \longrightarrow x_2 = 0.959299 \longrightarrow x_3 = 0.149888.$$

Capítol 3

Aplicacions de la Iteració

3.1 Càlcul de zeros de funcions

Suposem que volem calcular els zeros d'una determinada funció $F(x)$, és a dir, volem trobar les solucions de l'equació $F(x) = 0$. Si la funció $F(x)$ és continua i no tenim cap altra informació sobre els seus zeros, podem utilitzar el teorema de Bolzano per *localitzar-los* en el sentit de trobar un interval $[a, b]$ de manera que hi hagi un canvi de signe entre els seus extrems ($F(a)F(b) < 0$).

3.1.1 Mètode de bisecció

Un mètode molt senzill d'aplicació del teorema de Bolzano és el conegut amb el nom de *mètode de bisecció*. Si suposem que tenim un interval en el que hi ha un únic canvi de signe, es tracta de subdividir per la meitat l'interval inicial i quedar-nos amb aquella part que té un canvi de signe entre els seus extrems.

$$x_n = a, \quad x_{n+1} = b, \quad x_{n+2} = \frac{x_n + x_{n+1}}{2}$$

Si $F(x_n)F(x_{n+2}) < 0$ fem $x_{n+1} = x_{n+2}$, o en cas contrari, fem $x_n = x_{n+2}$. Amb això haurem descartat la meitat de l'interval i podem tornar a iterar.

La convergència del mètode està garantida per la continuïtat de la funció, però ja podem dir que el mètode de bisecció té una velocitat de convergència lineal (molt lenta) i per tant serà la última opció que prendrem.

3.1.2 Mètodes de punt fixe

La idea dels anomenats *mètodes de punt fixe* és passar de l'equació que volem resoldre $F(x) = 0$ a una del tipus $x = f(x)$ i resoldre-la per un esquema iteratiu.

Exemple 3.1.1 $F(x) = x^3 + x - 1000 = 0$; aleshores:

$$(i) \quad x = \underbrace{1000 - x^3}_{f_1(x)} \implies x_{n+1} = 1000 - x_n^3$$

$$(ii) \quad x = \underbrace{\frac{1000}{x^2} - \frac{1}{x}}_{f_2(x)} \implies x_{n+1} = \frac{1000}{x_n^2} - \frac{1}{x_n}$$

$$(iii) \quad x = \underbrace{\sqrt[3]{1000 - x}}_{f_3(x)} \implies x_{n+1} = \sqrt[3]{1000 - x_n}$$

3.1.2 Teorema del punt fix. Sigui $f : [a, b] \longrightarrow \mathbb{R}$ que satisfà:

$$(i) \quad f([a, b]) \subset [a, b]$$

1. f és de classe C^1 amb $|f'(x)| \leq L < 1, \forall x \in (a, b)$

Aleshores, $\forall x_0 \in [a, b]$, la successió definida per $x_{n+1} = f(x_n)$ convergeix cap a la solució s de l'equació $s = f(s)$ (punt fix).

Exemple 3.1.3 Si avaluem $F(9) = -262 < 0$ i $F(10) = 10 > 0$. Com a més tenim $F'(x) = x^2 + 1 > 0 \implies$ creixent \implies la solució és única.

$s \in [9, 10]$

$$|f'_1(x)| = |3x^2| > 243 > 243 > 1 \longrightarrow \text{no convergeix}$$

$$|f'_2(x)| = \frac{1}{x^2} \left| 1 - \frac{2000}{x} \right| > \frac{1}{81} |1 - 200| = 2.45 \dots > 1 \longrightarrow \text{no convergeix}$$

$$|f'_3(x)| = \frac{1}{3} |(1000 - x)^{-2/3}| = 0.003355 < 1 \longrightarrow \text{convergeix}$$

Proposició 3.1.4 L'error comés per un mètode de punt fix que aturem en el pas n -èsim es pot fitar per

$$\text{error} = |x_n - s| \leq \frac{L^n}{1 - L} |x_1 - x_0|.$$

Exemple 3.1.5 Volem saber quantes iteracions caldrien per conèixer s de l'exemple anterior $x_{n+1} = \sqrt[3]{1000 - x_n}$ amb error $< 10^{-6}$.

Si $x_0 \in [9, 10]$ veiem que se satisfan les hipòtesis del Teorema del punt fix $f([9, 10]) \subset [9, 10]$ (exercici)

Calculem ara

$$|f'(x)| < L < 1, \quad \forall x, \quad L = 0.003355$$

$$\text{volem que } \text{error} = |x_n - s| \leq \frac{L^n}{1 - L} |x_1 - x_0| = \frac{L^n}{1 - L} < 10^{-6}$$

$$n = 1 \implies 0.003366$$

$$n = 2 \implies 1.1293 \cdot 10^{-5}$$

$$n = 3 \implies 3.7891 \cdot 10^{-8}$$

o sigui

$$\begin{aligned}x_0 &= 9.5 \\x_1 &= 9 \\x_2 &= \dots \\x_3 &= 9.966666808.\end{aligned}$$

3.1.3 Acceleració de la convergència

Si denotem per $\varepsilon_n = x_n - s$, on s és un punt fix per $x = f(x)$ el número d'iteracions que caldrà perquè ε_n sigui *molt petit* pot arribar a ser *molt gran*.

Definició 3.1.6 Direm que un mètode iteratiu té ordre p si, i només si,

$$\lim_{h \rightarrow \infty} \frac{\varepsilon_{n+1}}{\varepsilon_n^p} = k, \quad \text{amb } k \neq 0 \text{ finit.}$$

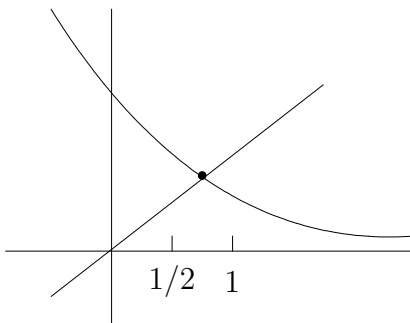
Generalment els mètodes iteratius són *d'ordre 1* (convergència lineal).

Es pot millorar la velocitat de convergència a partir del *mètode d'Aitken*, que calcula una successió de valors millorada de la forma

$$\bar{x}_n = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$$

Exemple 3.1.7 Volem calcular els zeros de la funció $F(x) = e^{-x} - x$. Passem al procés iteratiu $x = e^{-x} \implies f(x) = e^{-x}$. Tindrem

$$\begin{aligned}F(1/2) &= 0.106530 \\F(1) &= -0.6321\end{aligned} \implies s \in [1/2, 1]$$



$$f'(x) = -e^{-x} \implies |f'(x)| \leq e^{-1/2} < 1, \quad \forall x \in [1/2, 1]$$

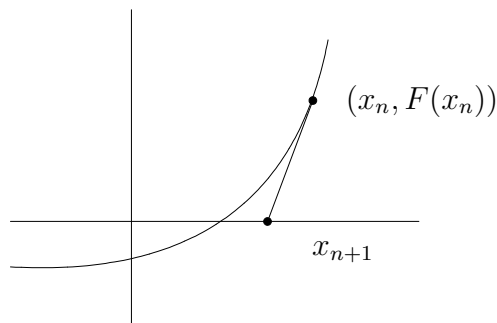
Iteració	Aitken
$x_0 = 0.5$	— — —
$x_1 = 0.606531$	— — —
$x_2 = 0.545239$	0.567624
$x_3 = 0.579703$	0.567299
$x_4 = 0.560056$	0.567193
$x_5 = 0.571172$	0.567193
$x_6 = 0.564863$	0.567159
$x_7 = 0.568438$	0.567148
$x_8 = 0.566410$	0.567145
$x_9 = 0.567560$	0.567144

3.1.4 Mètode de Newton

Es tracta de construir una iteració a partir de l'aproximació donada per la tangent a la funció $F(x)$ de la qual volem calcular els zeros, per això, tallem la recta tangent amb l'eix de les x :

$$\begin{cases} y - F(x_n) = F'(x_n)(x - x_n) \\ y = 0 \end{cases} \implies \boxed{x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}}$$

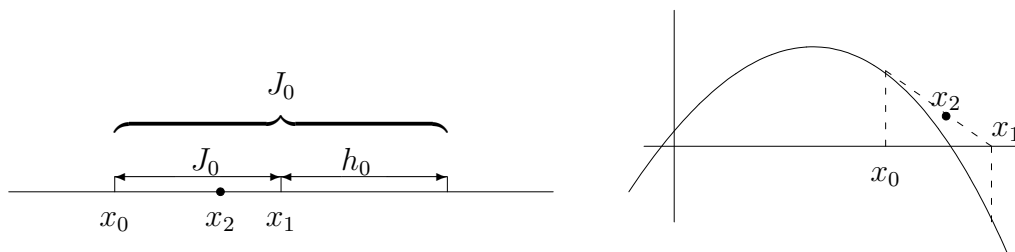
(= $f(x)$ una funció d'iteració)



Exemple 3.1.8 $F(x) = x - e^{-x} = 0 \implies F'(x) = 1 + e^{-x}$

$$\begin{aligned} x_0 = 0.5 & \longrightarrow f(x) = 0.106530 \\ x_1 = 0.56631100319721817 & \longrightarrow f(x) = -1.06 \cdot 10^{-1} \\ x_2 = 0.56714316503486217 & \longrightarrow f(x) = -1.30 \cdot 10^{-3} \\ x_3 = 0.56714329040978106 & \longrightarrow f(x) = -1.96 \cdot 10^{-7} \\ x_4 = 0.5671432904097838 & \longrightarrow f(x) = -4.40 \cdot 10^{-15} \end{aligned}$$

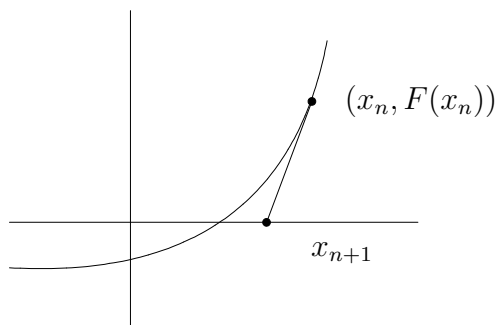
Proposició 3.1.9 Sigui $F : \mathbb{R} \longrightarrow \mathbb{R}$ i suposem $F'(x_0) \neq 0$. Si denotem per $h_0 = -\frac{F(x_0)}{F'(x_0)}$, $x_1 = x_0 + h_0$, $J_0 = [x_1 - |h_0|, x_1 + |h_0|]$, $M = \sup_{x \in J_0} |F''(x)|$. Llavors, si es satisfà $2 \left| \frac{F(x_0)M}{(F'(x_0))^2} \right| < 1$, aleshores el mètode de Newton amb valor inicial x_0 convergeix cap a la solució $F(x) = 0$ dins de J_0



Nota: La convergència del mètode de Newton és quadràtica.

3.1.5 Mètode de la secant (regular falsi)

La idea és utilitzar la recta que uneix els dos punts de la gràfica entre els que hi ha un canvi de signe



$$x_{x+1} = x_n - F(x_n) \frac{(x_n - x_{n-1})}{F(x_n) - F(x_{n-1})}$$

Observació 3.1.10 El mètode de la secant es pot pensar com una variació del mètode de Newton en el que aproximem

$$F'(x_n) \simeq \frac{F(x_n) - F(x_{n-1})}{x_n - x_{n-1}}.$$

Serà apropiat quan *no coneixem* $F'(x)$ o bé sigui difícil d'avaluar.

Nota: La convergència del mètode de la secant és d'ordre $\frac{1 + \sqrt{5}}{2} \simeq 1.618054$ (*Raó aurea o divina proporció*) i per tant està entre el mètode de punt fixe i el de Newton.

3.2 Sistemes lineals

Considerem un sistema d'equacions lineals:

$$\left. \begin{array}{l} 10x_1 + x_2 + x_3 = 12 \\ x_1 + 10x_2 + x_3 = 12 \\ x_1 + x_2 + 10x_3 = 12 \end{array} \right\} \Rightarrow \begin{pmatrix} 10 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ 12 \\ 12 \end{pmatrix}.$$

Normalment expressarem aquest sistema de la forma $Ax = b$.

Un mètode iteratiu de resolució d'aquest sistema d'equacions consisteix en expressar-lo de la forma

$$x^{(n+1)} = Bx^{(n)} + C.$$

Mètode de Jacobi

Es tracta d'aïllar la variable x_i de l'equació i -èsima

$$\left. \begin{array}{l} x_1 = \frac{1}{10}(12 - x_2 - x_3) \\ x_2 = \frac{1}{10}(12 - x_1 - x_3) \\ x_3 = \frac{1}{10}(12 - x_1 - x_2) \end{array} \right\} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{(n+1)} = \underbrace{\begin{pmatrix} 0 & -\frac{1}{10} & -\frac{1}{10} \\ -\frac{1}{10} & 0 & -\frac{1}{10} \\ -\frac{1}{10} & -\frac{1}{10} & 0 \end{pmatrix}}_B \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{(n)} + \underbrace{\begin{pmatrix} \frac{12}{10} \\ \frac{12}{10} \\ \frac{12}{10} \end{pmatrix}}_C.$$

Si prenem en aquest exemple

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow x^{(1)} = \begin{pmatrix} \frac{12}{10} \\ \frac{12}{10} \\ \frac{12}{10} \end{pmatrix} \Rightarrow x^{(2)} = \begin{pmatrix} 0.96 \\ 0.96 \\ 0.96 \end{pmatrix} \Rightarrow \dots \Rightarrow x^{(6)} = \begin{pmatrix} 0.999936 \\ 0.999936 \\ 0.999936 \end{pmatrix}.$$

Mètode de Gauss-Seidel

És la mateixa idea anterior, però utilitzant en cada equació el valor de les x_i calculades anteriorment

$$\left. \begin{array}{l} x_1^{(n+1)} = \frac{1}{10}(12 - x_2^{(n)} - x_3^{(n)}) \\ x_2^{(n+1)} = \frac{1}{10}(12 - x_1^{(n+1)} - x_3^{(n)}) \\ x_3^{(n+1)} = \frac{1}{10}(12 - x_1^{(n+1)} - x_2^{(n+1)}) \end{array} \right\} \Rightarrow \begin{cases} x_1^{(n+1)} = -\frac{1}{10}x_2^{(n)} - \frac{1}{10}x_3^{(n)} + \frac{12}{10} \\ \frac{1}{10}x_1^{(n+1)} + x_2^{(n+1)} = -\frac{1}{10}x_3^{(n)} + \frac{12}{10} \\ \frac{1}{10}x_1^{(n+1)} + \frac{1}{10}x_2^{(n+1)} + x_3^{(n+1)} = \frac{12}{10} \end{cases} \Rightarrow$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{10} & 1 & 0 \\ \frac{1}{10} & \frac{1}{10} & 1 \end{pmatrix}}_C \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{(n+1)} = \underbrace{\begin{pmatrix} 0 & -\frac{1}{10} & -\frac{1}{10} \\ 0 & 0 & -\frac{1}{10} \\ 0 & 0 & 0 \end{pmatrix}}_D \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{(n)} + \begin{pmatrix} \frac{12}{10} \\ \frac{12}{10} \\ \frac{12}{10} \end{pmatrix} \Rightarrow B = C^{-1}D.$$

Proposició 3.2.1 Si considerem la matriu del sistema $Ax = b$, $A = L + D + U$

$$A = \begin{pmatrix} \boxed{\begin{array}{c} \text{ } \\ \text{ } \\ \text{ } \end{array}} \\ \text{ } \\ \text{ } \end{pmatrix}$$

llavors les matrius que defineixen els mètodes iteratius $X^{n+1} = Bx^n + C$ de la forma

- *Jacobi*: $B_J = -D^{-1}(L + U)$ i $C_J = D^{-1}b$
- *Gauss-Seidel*: $B_{GS} = -(L + D)^{-1}U$ i $C_{GS} = (L + D)^{-1}b$

Convergència dels mètodes

Si considerem el mètode $x^{(n+1)} = Bx^{(n)} + C$ tindrem que la solució $x^{(n)}$ convergeix cap a un valor x (fixe per la iteració) si, i només si

radi espectral: $\rho(B) = \max_{1 \leq i \leq n} |\lambda_i(B)| < 1$ on $\lambda_1, \lambda_2, \dots$ són els valors propis de B .

Proposició 3.2.2 Si calculem la norma de la matriu d'iteració B llavors tindrem $\rho(B) \leq \|B\| < 1$ per a qualsevol norma matricial.

Definició 3.2.3 Recordem que les normes vectorials, essent $x \in \mathbb{R}^n$

$$\|x\|_1 = \sum_{i=1}^n |x_i| \implies \|B\|_1 = \max_{1 \leq j \leq n} \left(\sum_{i=1}^n |a_{ij}| \right) \quad (\text{màxim de les sumes per columnes})$$

$$\|x\|_\infty = \max_{i=1 \div n} |x_i| \implies \|B\|_\infty = \max_{1 \leq i \leq n} \left(\sum_{j=1}^n |a_{ij}| \right) \quad (\text{màxim de les sumes per files})$$

$$\|x\|_2 = \sqrt{x_1^2 + \dots + x_n^2} \implies \|B\|_2 = \sqrt{\rho(B^T B)}$$

En l'exemple que hem desenvolupat tindrem

$$B_J = \begin{pmatrix} 0 & -\frac{1}{10} & -\frac{1}{10} \\ -\frac{1}{10} & 0 & -\frac{1}{10} \\ -\frac{1}{10} & -\frac{1}{10} & 0 \end{pmatrix}$$

$$\|B_J\|_\infty = \frac{1}{5} < 1 \implies \text{convergent}$$

$$B_{GS} = -(L + D)^{-1}U \implies \|B_{GS}\| = \|(L + D)^{-1}\| \cdot \|U\| = \frac{\|U\|_\infty}{\|L + D\|_\infty} =$$

$$= \frac{2}{12} = \frac{1}{6} < 1 \implies \text{convergent}$$

3.2.1 Mètode de Newton per a sistemes d'equacions (no lineals)

Considerem ara el cas que volguem resoldre un sistema d'equacions no lineals, (escrivem el problema per a $n = 2$)

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases} \quad \text{o bé en forma matricial } F(\bar{x}) = 0, \text{ on } F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \bar{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Pel mètode de Newton tindriem l'expressió

$$\bar{x}_{n+1} = \bar{x}_n - DF^{-1}(\bar{x}_n)F(\bar{x}_n)$$

o bé en coordenades:

$$\begin{pmatrix} x_1^{(n+1)} \\ x_2^{(n+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(n)} \\ x_2^{(n)} \end{pmatrix} - \underbrace{DF^{-1}}_{\text{mat. } 2 \times 2} \begin{pmatrix} x_1^{(n)} \\ x_2^{(n)} \end{pmatrix} \begin{pmatrix} f_1(x_1^{(n)}, x_2^{(n)}) \\ f_2(x_1^{(n)}, x_2^{(n)}) \end{pmatrix}.$$

Exemple 3.2.4 Volem trobar una solució propera a $x_1 = 0.8$, $x_2 = 0.4$ del sistema d'equacions

$$\begin{cases} x - x^2 - y^2 = 0 \\ y - x^2 + y^2 = 0 \end{cases}$$

Si diem $F \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x - x^2 - y^2 \\ y - x^2 + y^2 \end{pmatrix}$, llavors $DF \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 - 2x & -2y \\ -2x & 1 + 2y \end{pmatrix}$ que, invertida

serà $DF^{-1} \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{\det(DF)} \begin{pmatrix} 1 + 2y & 2y \\ 2x & 1 - 2x \end{pmatrix}$ i $\det(DF) = 1 + 2(y - x) - 8xy$.

El mètode Newton serà:

$$\begin{pmatrix} x^{(n+1)} \\ y^{(n+1)} \end{pmatrix} = \begin{pmatrix} x^{(n)} \\ y^{(n)} \end{pmatrix} - \frac{1}{\det(DF)} \begin{pmatrix} 1 + 2y^{(n)} & 2y^{(n)} \\ 2x^{(n)} & 1 - 2x^{(n)} \end{pmatrix} \cdot \begin{pmatrix} x^{(n)} - x^{2(n)} - y^{2(n)} \\ y^{(n)} - x^{2(n)} + y^{2(n)} \end{pmatrix}.$$

Ara, si

$$\begin{pmatrix} x^{(0)} \\ y^{(0)} \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix} \implies \dots \implies \begin{pmatrix} 0.771844505 \\ 0.419643377 \end{pmatrix}.$$

3.3 Generació de nombres aleatoris

Volem descriure com generar de forma iterativa una successió de números aleatoris en un ordinador. Com es pot suposar tot procés controlat per un ordinador és totalment determinista i no és mai aleatori, per això per ser precisos caldria parlar de successions pseudoaleatòries.

Definició 3.3.1 Anomenarem generador de congruència lineal a la successió definida per l'operació

$$x_{n+1} = (ax_n + c) \bmod (m), \quad \text{on } a, c, m \in \mathbb{N}, \quad m > \max\{a, c\}.$$

Exemple 3.3.2 $x_{n+1} = (7x_n + 1) \bmod (10)$

$$x_0 = 8 \Rightarrow x_1 = 7, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 8$$

Observacions 3.3.3

1. El conjunt de valors x_n que es poden obtenir seran valors enters del conjunt $\{0, 1, \dots, m-1\}$.
2. El període màxim que pot tenir la successió obtinguda amb un generador de congruència lineal és m (en l'exemple és 4).
3. Per les dues observacions anteriors, ens convindrà treballar amb m gran i període màxim.

Definició 3.3.4 *Anomenarem generador de congruència multiplicativa al cas que $c = 0$, és a dir*

$$x_{n+1} = (ax_n) \bmod (m).$$

Observacions 3.3.5

4. No es pot utilitzar la llavor $x_0 = 0$.
5. Si $x_0 \neq 0$ tampoc voldrem que $x_0 = 0$ per a algun n , ja que llavors tots els valors serien 0. Que x_n sigui zero significa que ax_{n-1} és un múltiple de m però com $a < m$ això voldria dir $ax_{n-1} = m \Rightarrow a$ o x_{n-1} són divisors de m .

Per evitar això prendrem sempre m primer.

3.3.1 Generació de lleis uniformes $U([0, 1])$

Per obtenir valors uniformement distribuïts a l'interval $[0, 1]$ a partir de valors generats amb un mètode de congruència multiplicativa farem

$$\alpha_n = \frac{x_n}{m}, \quad \alpha_n \in [0, 1].$$

Notem que

- (i) α_n no prendrà mai el valor 1
- (ii) α_n només pot prendre valors $\left\{0, \frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}\right\}$, és a dir, no són valors continus (no uniformement distribuïts).

Exemple 3.3.6 $a = 7^5 = 16807$ $x_{n+1} = (ax_n) \bmod (m)$

$$m = 2^{31} - 1 = 2147483647 \quad (\text{és un nombre primer}).$$

Per implementar aquest mètode tindrem problemes d'emmagatzematge, ja que haurem de fer servir variables *long int* que podem prendre valors en el rang -2^{31} i $2^{31} - 1$. Amb això podem guardar m , però el producte ax_n pot superar aquest rang, aleshores cal fer:

Proposició 3.3.7 *Sigui $m = aq + r$ amb $q = \left\lfloor \frac{m}{a} \right\rfloor$ i $r = m \bmod (a)$. Considerem $x \in \{1, 2, \dots, m - 1\}$, llavors si $r < q$*

1. Els valors $a(x \bmod (m))$ i $r \left\lfloor \frac{x}{q} \right\rfloor$ estan compresos entre 0 i $m - 1$.
2. Definim $u = a(x \bmod (q)) - r \left\lfloor \frac{x}{q} \right\rfloor$, aleshores

$$(ax) \bmod (m) = \begin{cases} u & \text{si } u \geq 0 \\ u + m & \text{si } u < 0 \end{cases}$$

Mètode de la barreja

Una tècnica que es pot utilitzar per millorar qualsevol generador és el que s'anomena mètode de la barreja, que consisteix en alterar l'ordre en què apareixen els diferents nombres aleatoris. La idea de l'algorisme és la següent:

1. Generem en el moment inicial $k + 1$ valors aleatoris
2. Calculem un índex $j = \left\lfloor k \frac{P}{m} \right\rfloor$,

T_0	T_1	T_{k-1}
-------	-------	-------	-----------

 i $P = T_k$.
3. Fem $P = T_j$ (valor retornat).
4. Reomplim la taula amb $T_j = T_{k+1}$ el següent valor aleatori.
5. Retornem P .

3.3.2 Generació de lleis no uniformes

En aquest apartat veurem com a partir d'una $U([0, 1])$ podem generar altres lleis de distribució.

Llei exponencial

Si considerem que els valors α_n són valors obtinguts a partir d'una $U([0, 1])$ llavors la successió

$$e_n = m \ln(\alpha_n)$$

segueix una distribució exponencial de mitjana m (o de paràmetre $\frac{1}{m}$).

Llei normal

$$N\left(\underset{\substack{\uparrow \\ \text{mitjana}}}{0}, \underset{\substack{\uparrow \\ \text{variància}}}{1}\right)$$

$$\begin{cases} x_n = \cos(2\pi\alpha_{2n})\sqrt{-2\ln(\alpha_{2n-1})} \\ y_n = \sin(2\pi\alpha_{2n})\sqrt{-2\ln(\alpha_{2n-1})} \end{cases}$$

i la successió $x_1, y_1, x_2, y_2, x_3, y_3, \dots = \beta_n$.

Si es vol $N(\mu, \sigma^2) \implies \gamma_n = \sigma\beta_n + \mu$, on $\beta_n \in N(0, 1)$.

Capítol 4

Equacions diferencials ordinàries. Integració numèrica

Recordem que una equació diferencial ordinària (e.d.o.) és una equació en la què apareixen una certa funció i les seves derivades. L'ordre d'una e.d.o. és l'ordre de la derivada més alta que hi apareix.

4.1 E.d.o. de primer ordre

$$\frac{dx}{dt} = x' = f(t, x), \quad \text{on } f : [a, b] \times \mathbb{R} \longrightarrow \mathbb{R} \quad \text{és contínua} \quad (4.1)$$

$t \rightarrow$ variable independent, $x \rightarrow$ variable dependent $x = x(t)$.

Observació 4.1.1 Més endavant veurem que les e.d.o. d'ordre superior es poden estudiar com a *sistemes* d'equacions diferencials de *primer ordre*.

Definició 4.1.2 Direm que $u = u(t)$ és una solució de l'e.d.o. 4.1 si, i només si, és una funció diferenciable que satisfà l'equació, és a dir $u'(t) = f(t, u(t))$.

Nosaltres el que farem és calcular de forma numèrica (aproximada) les solucions d'aquestes e.d.o.'s. Més concretament resoldrem el que s'anomena *problema de Cauchy* (o de valor inicial)

$$\begin{cases} x' = f(x) & \text{e.d.o.} \\ x_0 = x(t_0) & \text{condició inicial} \end{cases}$$

Pel *teorema d'existència i unicitat* de solucions sabem que si f és de classe C^1 aleshores existeix una única solució que passa pel punt (t_0, x_0) .

4.2 Camp vectorial. Isoclines

Un camp vectorial o camp de pendent d'una equació diferencial $x' = f(t, x)$ és la direcció o pendent corresponent a cada punt (t, x) que ens ve donada per un petit vector amb

origen el punt (t, x) i inclinació (o pendent) igual a $f(t, x)$.

Observació 4.2.1 La solució $u(t)$ serà una corba *tangent* al vector anterior.

Exemple 4.2.2 Considerem l'equació diferencial $x' = -tx$ i mirem de representar el camp vectorial associat. Notem que

- (i) Per a $t = 0$ o $x = 0$ tenim $f(t, x) = 0 \Rightarrow$ vector horitzontal.
- (ii) Per a $t > 0$ o $x > 0$ els pendents són negatius i per a t fixat és més pendent quan x creix.
- (iii) Tenim simetria respecte els eixos i l'origen.

Definició 4.2.3 Les corbes de pendent constant s'anomenen isoclines $f(t, x) = k$.

La concentració d'*isoclines* ens dona les zones de màxim creixement de les solucions.

4.3 Càlcul de solucions

Mètodes d'un pas: Són els que el valor de la solució en un punt es calcula a partir del punt anterior.

Mètodes multipas: La solució en un punt es calcula a partir de més d'un dels punts anteriors.

Nosaltres estudiarem els mètodes d'un pas. La idea és la següent: volem calcular la solució de $x' = f(t, x)$ en un cert interval $[a, b]$ a partir de la condició inicial $x_0 = x(a)$ fins al valor $x = x(b)$.

Generalment, el que farem és dividir l'interval $[a, b]$ en N parts iguals

$$h = \frac{b - a}{N} \quad \text{i} \quad t_n = t_0 + nh, \quad n = 0 \div N$$

Mètode d'Euler

Si considerem la solució $x(t)$ que estem buscant prop de la condició inicial podem escriure

$$x(t) \simeq x(t_0) + x'(t_0)(t - t_0) = x_0 + f(t_0, x_0)(t - t_0)$$

de manera que si volem la solució en el següent punt $t_1 = t_0 + h$, llavors tindrem

$$x_1 = x(t_1) = x_0 + f(t_0, x_0)h.$$

Podem reiterar el procés per a $x_2 = x_1 + hf(t_1, x_1)$, ..., etc., i obtindrem

$$\begin{aligned} x_0 &= x(a) \\ x_{n+1} &= x_n + hf(t_n, x_n), \quad n = 0 \div N - 1. \end{aligned}$$

Exemple 4.3.1 Calcular la solució per a $t = 1$ del problema de Cauchy

$$\begin{cases} x' = x \\ x(0) = 1 \end{cases}$$

pel mètode d'Euler.

Observació 4.3.2 Per aquesta equació coneixem la solució

$$\frac{dx}{dt} = x \implies \frac{dx}{x} = dt$$

$\ln|x| = t + c$, $|x| = e^{t+c} = e^t \cdot e^c = \bar{k} \cdot e^t$, $\bar{k} > 0$, $x = k \cdot e^t$, $k \in \mathbb{R}$. Si $x(0) = 1 \implies 1 = k \cdot e^0 \implies k = 1 \implies \boxed{x(t) = e^t} \underset{t=1}{\implies} x(1) = e \simeq 2.7182$.

Fem-ho numèricament':

$$\begin{aligned} x_0 &= 1 \\ x_{n+1} &= x_n + h \cdot f(t_n, x_n) = x_n + h \cdot x_n = x_n(1 + h) \end{aligned}$$

Si fem $N = 2$ subdivisions: $h = \frac{b-a}{2} = 1/2$

$$\begin{aligned} x_0 &= 1 \\ x_1 &= x(1/2) = 1(1 + (1/2)) = 3/2 \\ x_2 &= x(1) = 3/2(1 + (1/2)) = 9/4 = 2.25 \end{aligned}$$

Si fem $N = 8$: $h = 1/8$

$$\begin{aligned} x_0 &= 1 \\ x_1 &= x(1/8) = 1(1 + (1/8)) = 1.12500 \\ x_2 &= x(2/8) = 1.26562 \\ x_3 &= x(3/8) = 1.42382 \\ &\dots \\ x_8 &= x(1) = 2.56578 \end{aligned}$$

Mètode d'Euler modificat

De fet el mètode d'Euler el podem pensar com fer passar una recta pel punt (t_n, x_n) el pendent de la qual és $m = f(t_n, x_n)$, és a dir, el valor de x' en aquest punt, $x_{n+1} = x_n + h \cdot m$. Una primera millora seria utilitzar el pendent en un punt més llunyà, per exemple el *punt mig* entre $[x_n, x_{n+1}]$, o sigui que seria el punt $\left(\frac{t_n + t_{n+1}}{2}, \frac{x_n + x_{n+1}}{2}\right)$, és a dir,

$$\begin{aligned} \frac{t_n + t_{n+1}}{2} &= \frac{t_n + (t_n + h)}{2} = \frac{2t_n + h}{2} = t_n + \frac{h}{2} \\ \frac{x_n + x_{n+1}}{2} &= \frac{x_n + (x_n + h \cdot f(t_n, x_n))}{2} = x_n + \frac{h}{2} f(t_n, x_n) \end{aligned}$$

o sigui que $x_{n+1} = x_n + h \cdot m = x_n + h \cdot f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}f(t_n, x_n)\right)$.

Exemple 4.3.3

$$\begin{cases} x' = x \\ x(0) = 1 \end{cases}$$

$$N = 2 \Rightarrow h = 1/2$$

$$x_0 = 1$$

$$x_1 = x(1/2) = x_0 + h \left(x_0 + \frac{h}{2}f(t_0, x_0) \right) = 1 + \frac{1}{2} \left(1 + \frac{1}{4} \cdot 1 \right) = 1 + \frac{5}{8} = \frac{13}{8} = 1.625$$

$$x_2 = x_1 + h \left(x_1 + \frac{h}{2} \cdot x \right) = x_1 \left(1 + h + \frac{h^2}{2} \right) = 2.64062$$

$$N = 8 \Rightarrow h = 1/8$$

$$x_0 = 1$$

$$x_1 = x\left(\frac{1}{8}\right) = 1.13281$$

$$x_2 = 1.28326$$

⋮

$$x_8 = x(1) = 2.71184$$

Mètode d'Euler enrera

Considerem ara:

$$x_n = x(t_n) = x(t_{n+1} - h) \cong x(t_{n+1}) - x'(t_{n+1})h = x_{n+1} - hf(t_{n+1}, x_{n+1}) :$$

$x_{n+1} = x_n + hf(t_{n+1}, x_{n+1})$, equació no lineal (si f ho és) en x_{n+1} que s'ha de resoldre amb algun mètode de càlcul de zeros de funcions: Newton, Secant, etc. i trobar x_{n+1} .

Exemple 4.3.4

$$\begin{cases} x' = x \\ x(0) = 1 \end{cases}$$

$$x_{n+1} = x_n + hx_{n+1} \iff x_{n+1} = \frac{1}{1-h}x_n ;$$

$$\begin{aligned}
N = 2 &\Rightarrow h = 1/2, & \frac{1}{1-h} &= 2 \\
x_0 &= 1 \\
x_1 &= 2 \\
x_2 &= 4 \\
N = 8 &\Rightarrow h = 1/8, & \frac{1}{1-h} &= \frac{8}{7} \\
x_0 &= 1 \\
x_1 &= \frac{8}{7} = 1.14286 \\
&\vdots \\
x_8 &= 2.91028
\end{aligned}$$

Mètodes de Runge-Kutta

Es tracta de tota una família de mètodes (que veurem després amb més detall) pels quals el que fem és utilitzar un pendent obtingut a partir d'una mitjana ponderada de pendents. El més conegut és el mètode de Runge-Kutta 4 que descriurem de la forma següent

$$x_{n+1} = x_n + h \cdot m_{\text{RK4}}$$

on

$$\begin{aligned}
m_1 = f(t_n, x_n) &\longrightarrow \text{pendent al } \textit{punt inicial} \\
m_2 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}m_1) &\longrightarrow \text{pendent al } \textit{punt mig} \text{ del segment de pendent } m_1 \\
m_3 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}m_2) &\longrightarrow \text{al mateix per un segment de pendent } m_2 \\
m_4 = f(t_n + h, x_n + h \cdot m_3) &\longrightarrow \text{pendent al } \textit{final} \text{ d'un segment amb pendent } m_3
\end{aligned}$$

$$\text{finalment } m_{\text{RK4}} = \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4).$$

Exemple 4.3.5

$$\begin{cases} x' = x \\ x(0) = 1 \end{cases}$$

$$\begin{aligned}
N = 2 &\longrightarrow x_2 = x(1) = 2.71734 & \text{Sol.: } e = 2.718282 \\
N = 8 &\longrightarrow x_p = x(1) = 2.718277
\end{aligned}$$

Anàlisi de l'error

En aquests mètodes d'integració d'equacions diferencials ordinàries tenim dues fonts principals d'error:

- *error de discretització o de truncament*, generat pel fet que utilitzen un algorisme aproximat per calcular la solució.

- *error d'arrodoniment*, generat pel fet que totes les operacions es fan amb un nombre finit de dígitos.

Aquestes dues fonts d'error fan que l'expressió de l'error comès en funció del pas triat h tingui un comportament del tipus

L'únic tipus d'error que podem millorar és l'error de truncament que podem definir com $\varepsilon_n = |\bar{x}(t_n) - x_n|$ on $\bar{x}(t)$ és la solució exacta.

Definició 4.3.6 *Direm que un mètode d'integració és d'ordre h^p , $\mathcal{O}(h^p)$ si, i només si, existeix $h_q > 0$ o $k > 0$ tal que $\varepsilon_n \leq k \cdot h^p$ on $h \in [0, h_q]$.*

Si $p \geq 1$ llavors $\lim_{h \rightarrow 0} \varepsilon_n = 0$ i direm que el mètode és convergent de fet, mentre més gran sigui p , més ràpida és la convergència.

Així, per exemple, si un cert mètode satisfà $\varepsilon_n \leq h^p$ ($k = 1$), aleshores si $h = 0.01$ i

$$\begin{aligned} p = 1 &\longrightarrow \varepsilon_n \leq 0.01 \\ p = 2 &\longrightarrow \varepsilon_n \leq 0.0001 \\ p = 3 &\longrightarrow \varepsilon_n \leq 0.000001 \end{aligned}$$

Proposició 4.3.7 *Si escrivim de forma genèrica un mètode d'integració de la forma $x_{n+1} = x_n + h \cdot \Phi(t_n, x_n, h)$ tindrem que si $\left| \frac{x(t+h) - x(t)}{h} - \Phi(t_n, x(t), h) \right| = \mathcal{O}(h^p)$ llavors el mètode és d'ordre p .*

Exemple 4.3.8 :

1. El mètode d'Euler té ordre 1:

$$\Phi(t, x, h) = f(t, x(t)) \quad x(t+h) = x(t) + x'(t)h + x''(t)\frac{h^2}{2} + \dots$$

$$\left| \frac{x(t+h) - x(t)}{h} - f(t, x(t)) \right| = \left| \left(x'(t) + x''(t)\frac{h}{2!} + \dots \right) - f(t, x(t)) \right| = \mathcal{O}(h)$$

2. El mètode d'Euler modificat té ordre 2:

$$f\left(t + \frac{h}{2}, x + \frac{h}{2}f(t, x)\right) = f(t, x) + f_t(t, x)\frac{h}{2} + f_x(t, x)\frac{h}{2}f(t, x) + \mathcal{O}(h^2)$$

aleshores tindrem

$$\begin{aligned} \left| \frac{x(t+h) - x(t)}{h} - f\left(t + \frac{h}{2}, x + \frac{h}{2}f(t, x)\right) \right| &= \left| x'(t) + x''(t)\frac{h}{2} + \dots \right. \\ &\quad \left. - \left(f(t, x) + f_t(t, x)\frac{h}{2} + f_x(t, x)\frac{h}{2}f(t, x) \right) \right| \end{aligned}$$

i com $x' = f(t, x) \Rightarrow x'' = \frac{dx'}{dt} = \frac{d}{dt}f(t, x(t)) = f_t + f_x \frac{dx}{dt} = f_t + f_x f$, llavors $x''\frac{h}{2} = \frac{h}{2}(f_t + f_x f) \Rightarrow \text{error} \cong \mathcal{O}(h^2)$.

Mètodes de Runge-Kutta

Els mètodes de Runge-Kutta són una família de mètodes que es basen en construir fórmules de manera que el mètode resultant tingui un ordre predeterminat.

Esquema general:

$$\begin{aligned}x_0 &= x(a) \\x_{n+1} &= x_n + h \sum_{i=1}^k c_i k_i^{(n)}\end{aligned}$$

on $k \in \mathbb{N}$ està prefixada, les $c_i \in \mathbb{R}$ són constants a determinar i les $k_i^{(n)} = k_i^{(n)}(t_n, x_n, h)$ són funcions definides recurrentment per

$$\begin{aligned}k_1^{(n)} &= f(t_n, x_n) \\k_i^{(n)} &= f\left(t_n + \alpha_i h, x_n + h \sum_{j=1}^{i-1} \beta_{ij} k_j^{(n)}\right), \quad i = 2 \div k\end{aligned}$$

amb α_i, β_{ij} constant a determinar de manera que el mètode sigui de l'ordre desitjat.

– RK-1 \Rightarrow mètode d'Euler

– RK-2 $\Rightarrow x_{n+1} = x_n + h\Phi(t, x, h)$

on $\Phi(t, x, h) = c_1 k_1^{(1)} + c_2 k_2^{(2)} = c_1 f(t, x) + c_2 f(t + a_2 h, x + hb_2, f(t, x))$. Si prenem $c_1 = c_2 = 1/2$ i $a_2 = b_2 = 1 \Rightarrow$ ordre 2 (exercici)

$$x_{n+1} = x_n + h \left(\frac{1}{2} f(t_n, x_n) + \frac{1}{2} f(t_n + h, x_n + hf(t_n, x_n)) \right)$$

– RK-4 \Rightarrow ja està vist amb $m_i = k_1^{(4)}$.

4.3.1 Integració autòmatica. Control de pas

Es tracta de trobar un algorisme que adapti el pas d'integració de forma automàtica a les "condicions" del camp vectorial. La idea és fer el càlcul de dues formes i veure que no són gaire diferents. Veiem-ho en detall.

Mètode de Runge-Kutta-Fehlberg (RKF-45)

Tal com hem dit es tracta de fer un càlcul amb un RK4 i també amb un RK5, de manera que podem anar modificant el pas en relació a les diferències obtingudes.

Per a això definirem unes constants adequades:

$$\begin{aligned}
 k_1 &= f(t_n, x_n) \\
 k_2 &= f\left(t_n + \frac{1}{4}h_n, x_n + \frac{h_n}{4}k_1\right) \\
 k_3 &= f\left(t_n + \frac{3}{8}h_n, x_n + h_n\left(\frac{3}{32}k_1 + \frac{9}{32}k_2\right)\right) \\
 k_4 &= f\left(t_n + \frac{12}{13}h_n, x_n + h_n\left(\frac{1931}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right)\right) \\
 k_5 &= f\left(t_n + h_n, x_n + h_n\left(\frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right)\right) \\
 k_6 &= f\left(t_n + \frac{1}{2}h_n, x_n + h_n\left(-\frac{8}{71}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right)\right)
 \end{aligned}$$

i a partir d'aquestes constants calculem les aproximacions

$$\begin{aligned}
 \text{RK4: } \bar{x}_{n+1} &= x_n + h_n \left[\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \right] \\
 \text{RK5: } \hat{x}_{n+1} &= x_n + h_n \left[\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \right]
 \end{aligned}$$

i tindrem una estimació de l'error comès a partir de la diferència $\|\hat{x}_{n+1} - \bar{x}_{n+1}\|$.

Veiem com obtenim la fórmula del *pas òptim*, si suposem

$$\begin{aligned}
 \bar{x}_{n+1} &= x_n + h_n \Phi_4(t_n, x_n, h_n) \\
 \hat{x}_{n+1} &= x_n + h_n \Phi_5(t_n, x_n, h_n)
 \end{aligned}$$

tindrem que $\|\hat{x}_{n+1} - \bar{x}_{n+1}\| \cong \|h_n(\Phi_4 - \Phi_5)\|$. Recordem que com

$$\begin{aligned}
 \text{RK4 té ordre 4} &\implies \Phi_4 \cong W_4(t_n)h_n^4 \\
 \text{RK5 té ordre 5} &\implies \Phi_5 \cong W_5(t_n)h_n^5
 \end{aligned}$$

d'on

$$\|\hat{x}_{n+1} - \bar{x}_{n+1}\| \cong W_4(t_n)h_n^5. \quad (4.2)$$

Si suposem que el càlcul s'ha fet amb un error menor que *tol* llavors voldrem predir el nou pas h_{n+1} de manera que

$$\|\hat{x}_{n+2} - \bar{x}_{n+2}\| \cong W_4(t_{n+1})h_{n+1}^5 \leq \text{tol}.$$

Si considerem $W_4(t_{n+1}) = W_4(t_n + h_{n+1}) \stackrel{\text{Taylor}}{\cong} W_4(t_n) + C(h_{n+1})$, tindrem $W_4(t_n)h_{n+1}^5 \leq \text{tol}$

$$h_{n+1}^5 \leq \frac{\text{tol}}{W_4(t_n)} \stackrel{(4.2)}{\cong} \frac{\text{tol}}{\|\hat{x}_{n+1} - \bar{x}_{n+1}\|} h_n^5 \implies |h_{n+1}| = |h_n| \sqrt[5]{\frac{\text{tol}}{\|\hat{x}_{n+1} - \bar{x}_{n+1}\|}}.$$

Observacions 4.3.9

1. Donat que en cada cas només es fa una correcció del pas, haurem de donar un *pas inicial* més aviat *petit* per garantir que el primer càlcul és correcte.
2. Si volem forçar un pas d'integració constant, llavors caldrà fer $h = h_{min} = h_{max}$.
3. Si volem integrar fins a un valor fixat t_{fin} exacte, llavors caldrà fer la integració fins que $t > t_{fin}$ i llavors fer un últim pas prenent $h = t_{fin} - t$ (serà negatiu).

Utilització de l'integrador RK45F

Distingirem les següents parts en la utilització d'aquest paquet

- programa principal (main)
- acció que conté les equacions diferencials
- paquet RK45F
- paquet gràfics (opcional)

1. Les equacions diferencials les escriurem en forma vectorial

```
void camp (double t, double x[], int n, double y[])
```

$$\begin{cases} \dot{x} = y + x(\mu - x^2 - y^2) \\ \dot{y} = -x + y(\mu - x^2 - y^2) \end{cases} \iff \begin{cases} y[0] = x[1] + x[0](\mu - x[0] * x[0] - x[1] * x[1]); \\ y[1] = -x[0] + x[1](\mu - x[0] * x[0] - x[1] * x[1]) \end{cases}$$

$t \rightarrow$ valor actual de la variable independent

$x[] \rightarrow$ vector que conté el punt actual

$n \rightarrow$ dimensió del sistema

$y[] \rightarrow$ vector de derivades

2. Programa principal per utilitzar el RK45F

(i) Inicialització

– Paràmetres inicials:

$n \rightarrow$ dimensió del sistema d'equacions diferencials

$h \rightarrow$ pas actual d'integració ($t_{n+1} = t_n + h$)

$h_{min}, h_{max} \rightarrow$ valors mínim i màxim permesos per h

$tol \rightarrow$ error màxim que volem cometre.

– Condicions inicials

$t = 0 \rightarrow$ valor inicial de la variable independent de l'òrbita

$$\left. \begin{array}{l} x[0] = \dots \\ x[1] = \dots \\ \dots \\ x[n-1] = \dots \end{array} \right\} \text{condició inicial}$$

– Inici del RK

ini_rk45F(n).

(ii) Seguiment d'una òrbita. Generalment és un bucle de crides al RK de manera que anem integrant les equacions diferencials fins que es doni el que anomenarem *condició de parada* i que pot dependre de: $\left\{ \begin{array}{l} \text{temps final} \\ \text{valor d'alguna variable} \\ \text{crides del RK} \end{array} \right.$

(iii) Finalització. Alliberem el RK \rightarrow *fi_rk45F(n)*;

3. Paquet RK45F.C

La crida al mètode RK45F (en la nostra implementació) és la següent:

```
double rk45f(double *t, double x[], double *h, double tol, double hmin,
             double hmax, int n, void (*deriv)(double, double *, int, double *))
```

on els paràmetres són:

- **t**: variable independent (temps). *Entrada*: temps corresponent al punt actual. *Sortida*: nou valor corresponent al següent punt.
- **x**: variable dependent (posició). *Entrada*: posició corresponent al temps actual. *Sortida*: nova posició corresponent al següent instant de temps.
- **h**: pas temporal (pot ser modificat per la rutina d'acord amb la tolerància.)
- **tol**: tolerància per controlar l'error d'integració.
- **hmin**: mínim pas temporal permès.
- **hmax**: màxim pas temporal permès.
- **n**: dimensió del sistema d'equacions diferencials.
- **deriv**: funció que avalua el camp vectorial.

Exemple 4.3.10 Integrar les equacions de Lorenz

$$\begin{array}{ll} x' = s(-x + y) & s = 10.0 \\ y' = rx - y - xz & b = 8/3 \\ z' = -bz + xy & r = 28.0 \end{array}$$

p.i. (1.5, -1.0, 0), finestra (-30, 30, -30, 30).

4.3.2 Retrat de Fase d'un sistema d'equacions diferencials ordinàries. Objectes invariants

S'anomena retrat de fase d'un sistema d'equacions diferencials ordinàries per exemple $x' = f(x, y)$, $y' = g(x, y)$, a la representació de les òrbites en els eixos x , y .

Observació 4.3.11 En aquesta representació perdem informació sobre com es recorren les trajectòries respecte del temps.

Objectes invariants

Són conjunt invariants pel fluxe i que constitueixen l'esquelet del retrat de fases. Podríem dir que coneixent aquests objectes i el seu caràcter (estable o inestable, etc.) ja tenim determinat el comportament de la resta de trajectòries.

Punts d'equilibri

Són punts fixes pel fluxe i es caracteritzen per satisfer $x' = 0$. Si $x' = f(x)$ llavors els punts d'equilibri satisfan $f(x) = 0$.

El caràcter estable o inestable dels punts d'equilibri està relacionat amb els valors propis de la matriu diferencial, Df en el punt d'equilibri, $\lambda = 0 \Rightarrow$ inestabilitat i $\lambda < 0 \Rightarrow$ estabilitat.

Òrbites periòdiques

Són trajectòries que satisfan $x(t + T) = x(t)$, on T s'anomena període de l'òrbita (són trajectòries tancades).

Observació 4.3.12 En dim-2 les òrbites periòdiques separen l'espai de fases.

Objectes invariants en $\dim \geq 3$

4.3.3 Càlcul numèric d'òrbites periòdiques. Secció de Poincaré

Suposem que tenim un sistema 2-dim i que existeix una òrbita periòdica (atractora) que volem calcular numèricament utilitzant el RK45F. En primer lloc el que farem és reduir la dimensió del sistema passant a estudiar el que s'anomena *aplicació de Poincaré*.

Donat un sistema

$$\begin{cases} x' = f(x, y) \\ y' = g(x, y) \end{cases}$$

definirem una superfície de secció com una certa funció $h(x, y) = 0$. Generalment $x - \text{ctant} = 0$, que escriurem com $x = x_{\text{secc}}$ i l'aplicació de Poincaré associada com la

que donat un punt $(x(t_i), y(t_i))$ sobre $x = x_{secc}$ dona el següent punt de tall amb la secció $(x(t_f), y(t_f))$.

Quan intentem calcular el següent punt de l'òrbita sobre la secció de Poincaré amb un mètode amb control automàtic de pas tenim que distingir dos etapes:

1. Determinació del tall de la secció mitjançant els signes del $x - x_{secc}$.
2. Càlcul del punt de tall \Rightarrow Newton.

El punt de tall el caracteritzen com que satisfà $x(t) - x_{secc} = 0$, és a dir, si $F(t) = x - x_{secc}$ volem $F(t) = 0$ i ho fem per Newton

$$t_{n+1} = t_n - \frac{F(t_n)}{F'(t_n)} \Rightarrow \underbrace{t_{n+1} - t_n}_h = -\frac{F(t_n)}{F'(t_n)}$$

i com $F(t_n) = x(t_n) - x_{secc} \Rightarrow F'(t_n) = x'(t_n)$; de les equacions diferencials, llavors triarem com a pas

$$h = -\frac{(x[t_n] - x_{secc})}{x'[x_n]} \Leftrightarrow \left(h = -\frac{(x[0] - x_{secc})}{y[0]} \text{ en llenguatge C} \right)$$

repetirem el procés fins que

$$|x[0] - x_{secc}| < 10^{-12} \quad (\text{per exemple } \dots)$$

Pel que fa a l'òrbita periòdica serà aquell punt de la superfície de secció que sigui fixa per l'aplicació de Poincaré

$$x(t_i) = x(t_f) \implies (|x(t_f) - x(t_i)| < 10^{-10})$$

Nota. El període coincidirà amb $t_i - t_f \equiv T$ quan ja estem sobre l'òrbita periòdica.

4.3.4 Models d'interacció

Anem a estudiar el comportament qualitatiu dels sistemes anomenats d'interacció que, generalment s'apliquen a models de creixement d'espècies, models de competició, de combat, etc.

Model de Malthus

Malthus era un economista britànic que, sobre l'any 1800, va donar un model per descriure el creixement d'una població. La *Llei de Malthus* estableix que la velocitat de creixement d'una determinada espècie és proporcional a la quantitat d'individus existents a cada moment.

Si $x(t)$ és el número d'individus en un cert instant t , llavors

$$\frac{dx}{dt} = K \cdot x(t) \implies x(t) = x_0 e^{k(t-t_0)}$$

$k > 0 \Rightarrow$ creixement, $k < 0 \Rightarrow$ decreixement.

Veiem que la llei de Malthus es correspon amb un tipus de creixement exponencial \Rightarrow poc realista.

Model logístic

La ideal del model logístic és considerar que la constant de creixement no és sempre constant, sinó que dependrà també del nombre d'individus, és a dir, $k(t) = a - b \cdot x(t)$ o sigui que la llei logística es pot escriure com

$$\frac{dx}{dt} = (a - bx)x \quad \text{o bé} \quad \frac{dx}{dt} = kx \left(1 - \frac{x}{s}\right) \quad \text{amb } k = a, \quad s = a/b$$

Notem que si x és petit, llavors $1 - \frac{x}{s} \cong 1 \Rightarrow$ llei de Malthus.

La solució d'aquesta equació diferencial ordinària és

$$x(t) = \frac{s}{1 + Ce^{-kt}}, \quad \text{on } C = e^{kt_0} \left(\frac{s}{x_0} - 1 \right), \text{ si } x(t_0) = x_0 \text{ c.i.}$$

el comportament d'aquestes solucions és de la forma

Observació 4.3.13 Podria modificar-se el model considerant que el nivell de saturació podria dependre del temps $s = s(t)$ per considerar nivells de saturació lligats a les estacions, a la llum solar, etc.

Exemples 4.3.14 (I) Model depredador-presa:

$x_1 \rightarrow$ població de preses

$x_2 \rightarrow$ població de depredadors

De fet, podrien ser qualsevol tipus de problema en què x_2 necessita de x_1 per existir i augmentar, mentre que x_1 subsisteix i augmenta per si sol.

Les equacions d'aquest model són

$$\begin{cases} x_1' = k_1 x_1 \left(1 - \frac{x_1}{s_0}\right) - \alpha_1 x_1 x_2 \\ x_2' = -k_2 x_2 + \alpha_2 x_1 x_2 \end{cases}$$

on k_i, s_i, α_i són constant positives.

Notem que x_1 segueix un creixement logístic i x_2 decreixement malthusià. Els coeficients α_1, α_2 mesuren el grau d'aprofitament; així α_2 mesura el que s'anomena "eficiència" del depredador.

(II) Model de competició.

Si considerem ara dues poblacions x_1 , x_2 que competeixen per la mateixa font d'alimentació, tindrem

$$\begin{cases} x'_1 = k_1 x_1 \left(1 - \frac{x_1}{s_1}\right) - \alpha_1 x_1 x_2 \\ x'_2 = k_2 x_2 \left(1 - \frac{x_2}{s_2}\right) - \alpha_2 x_1 x_2 \end{cases}$$

La relació entre α_1 i α_2 determina la relació d'eficiència entre x_1 i x_2 .

(III) Model de combat.

Anàlogament es pot plantejar un model de combat entre militars en situació de guerrilla (sense reforços)

$$\begin{cases} x'_1 = -k_1 x_1 - \alpha_1 x_1 x_2 \\ x'_2 = -k_2 x_2 - \alpha_2 x_1 x_2 \end{cases}$$

Estudi qualitatiu d'un model depredador-presa

Veiem com estudiar de forma "qualitativa" el comportament d'un model depredador-presa. Fixem $s = 2$, $k_1 = k_2 = \alpha_1 = \alpha_2 = 1$:

$$\begin{cases} x'_1 = k_1 x_1 \left(1 - \frac{x_1}{s}\right) - \alpha_1 x_1 x_2 \equiv x_1 \left(1 - \frac{x_1}{2}\right) - x_1 x_2 \\ x'_2 = -k_2 x_2 + \alpha_2 x_1 x_2 \equiv -x_2 + x_1 x_2 \end{cases}$$

Punts d'equilibri

Hem d'imposar $x'_1 = 0$, $x'_2 = 0$

$$\begin{cases} x_1 = \left[k_1 \left(1 - \frac{x_1}{s}\right) - \alpha_1 x_2 \right] = 0 \\ x_2 = [-k_2 + \alpha_2 x_1] = 0 \end{cases} \implies \begin{cases} x_2 = 0 \implies \begin{cases} x_1 = 0 \\ x_1 = s \end{cases} \\ x_1 = \frac{k_2}{\alpha_2} = 1 \implies x_2 = \frac{k_1 \left(1 - \frac{k_2}{s}\right)}{\alpha_1} = \frac{1}{2} \end{cases}$$

Tres punts d'equilibri $p_1 = (0, 0)$, $p_2 = (s, 0)$, $p_3 = (1, 1/2)$.

Isoclines (de velocitat zero)

$$\begin{aligned} x_1 = 0 &\rightarrow \text{eix } x_2 \\ x'_1 = 0 &\implies x_2 = \frac{1}{\alpha_1} \left(k_1 \left(1 - \frac{x_1}{s}\right) \right) \rightarrow \text{recta } x_2 = a + b x_1 \\ &= \frac{k_1}{\alpha_1} - \frac{k_1 x_1}{\alpha_1 s} \equiv 1 - \frac{x_1}{2} \\ x_2 = 0 &\rightarrow \text{eix } x_1 \\ x'_2 = 0 &\implies x_1 = \frac{k_2}{\alpha_2} = 1 \end{aligned}$$

Caràcter dels punts d'equilibri

Matriu diferencial:

$$DF = \begin{pmatrix} \frac{\partial x'_1}{\partial x_1} & \frac{\partial x'_2}{\partial x_1} \\ \frac{\partial x'_1}{\partial x_2} & \frac{\partial x'_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} k_1 - 2\frac{k_1}{s} & -\alpha_1 \\ \alpha_2 x_1 & -k_2 + \alpha_2 x_1 \end{pmatrix}$$

$$DF \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} k_1 & 0 \\ 0 & -k_2 \end{pmatrix}$$

$$DF \begin{pmatrix} s \\ 0 \end{pmatrix} = \begin{pmatrix} -k_1 & -\alpha_1 s \\ \alpha_2 s & -k_2 + \alpha_2 s \end{pmatrix} = \begin{pmatrix} -1 & -s \\ s & -1 + s \end{pmatrix} =$$

$$= \begin{pmatrix} -1 & -2 \\ 2 & 1 \end{pmatrix} \begin{array}{l} \nearrow \det > 0 \\ \searrow t_2 = 0 \end{array} \lambda^2 - 0\lambda + 3 = 0 \Rightarrow \lambda = \pm\sqrt{31} \rightarrow \text{focus}$$

$$DF \begin{pmatrix} ? \\ 1/2 \end{pmatrix} = \begin{pmatrix} -1/2 & -1 \\ ? & ? \end{pmatrix} \begin{array}{l} \nearrow \det > 0 \\ \searrow t_2 < 0 \end{array} \rightarrow \text{sella}$$

Capítol 5

Apendix A: Implementació en llenguatge C del mètode RK45F

A continuació passem el codi de la implementació de l'integrador d'equacions diferencials basat en el mètode de Runge Kutta d'ordre 4 i 5 amb control automàtic de pas:

```
/*-----  
* Aquest paquet serveix per integrar equacions diferencials ordinaries  
* amb el metode de Runge-Kutta Felbergh d'ordres 4 i 5 i control  
* automatic del pas d'integracio.  
* (c) Toni Susin  
-----*/  
  
#include <math.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
void ini_rk45f(int n);  
void fi_rk45f(int n);  
double rk45f(double *t, double x[], double *h, double tol,  
             double hmin, double hmax, int n,  
             void (*deriv)(double, double *, int, double *));  
  
static double alfa[6]={  
    0.e0,          1.e0/4.e0,          3.e0/8.e0,  
    12.e0/13.e0,  1.e0,              1.e0/2.e0};  
  
static double beta[16]={  
    0.e0,          1.e0/4.e0,          3.e0/32.e0,          9.e0/32.e0,  
    1932.e0/2197.e0, -7200.e0/2197.e0,  7296.e0/2197.e0,  439.e0/216.e0,  
    -8.e0,         3680.e0/513.e0,     -845.e0/4104.e0,     -8.e0/27.e0,
```

```

        2.e0,      -3544.e0/2565.e0,   1859.e0/4104.e0,   -11.e0/40.e0};

static double c4[6]={
    25.e0/216.e0,      0.e0,      1408.e0/2565.e0,
    2197.e0/4104e0,   -1.e0/5.e0,      0.e0};

static double c5[6]={
    16.e0/135.e0,      0.e0,      6656.e0/12825.e0,
    28561.e0/56430.e0,   -9.e0/50.e0,      2.e0/55.e0};

static double *x4,*x5,*x_pond,*dx,*k[6];
static int neq=0;

#define MAX(a,b) (((a)<(b)) ? (b) : (a))
#define SGN(a) (((a)<0) ? -1 : 1)

void ini_rk45f(int n)
/*
aquí reservem l'espai de memòria pel paquet. S'ha de cridar aquesta acció
abans de cridar el propi integrador rk45f.

parametres:
n: dimensió del sistema d'equacions que volem integrar.
*/
{
    int j;
    if (n < 1) {puts("ini_rk45f: n ha de ser 1 com a mínim"); exit(1);}
    if (neq != 0)
        {
            free(x4);
            free(x5);
            free(x_pond);
            free(dx);
            for (j=0; j<6; j++) free(k[j]);
        }
    neq=n;
    x4=(double*)malloc(n*sizeof(double));
    if (x4 == NULL) {puts("ini_rk45f: falta memòria (1)"); exit(1);}
    x5=(double*)malloc(n*sizeof(double));
    if (x5 == NULL) {puts("ini_rk45f: falta memòria (2)"); exit(1);}
    x_pond=(double*)malloc(n*sizeof(double));
    if (x_pond == NULL) {puts("ini_rk45f: falta memòria (3)"); exit(1);}
    dx=(double*)malloc(n*sizeof(double));

```

```

    if (dx == NULL) {puts("ini_rk45f: falta memoria (4)"); exit(1);}
    for (j=0; j<6; j++)
    {
        k[j]=(double*)malloc(n*sizeof(double));
        if (k[j] == NULL) {puts("ini_rk45f: falta memoria (5)"); exit(1);}
    }
    return;
}

void fi_rk45f(int n)
/*
alliberem la memoria reservada per ini_rk45f.

parametre:
n: dimensio del sistema d'equacions diferencials, ha de coincidir amb el
    valor previament utilitzat a ini_rk45f.
*/

{
    int j;
    if (n != neq) puts("end_rk45f warning: dimensions do not coincide!");
    free(x4);
    free(x5);
    free(x_pond);
    free(dx);
    for (j=0; j<6; j++) free(k[j]);
    neq=0;
    return;
}

double rk45f(double *t, double x[], double *h, double tol,
             double hmin, double hmax, int n,
             void (*deriv)(double, double *, int, double *))
/*
Aquest es propiament l'integrador rk45f, de fet només fem un pas
d'integració en cada crida. El punt inicial (t,x) es canvia pel
nou punt sobre la mateixa orbita.
L'error es sempre menor que la tolerancia fixada per tol. Una estimació
de l'error comes en el pas actual es el valor de retorn de la funcio.

parametres:
t:      temps. entrada: temps corresponent al punt actual.
        sortida: nou valor corresponent al següent punt.

```

```

x:      posicio. els mateixos comentaris que pel temps.
h:      pas temporal (pot ser modificat per la rutina d'acord amb la
        tolerancia.)
tol:    tolerancia per controlar l'error d'integracio.
hmin:   minim pas temporal permes.
hmax:   maxim pas temporal permes.
n:      dimensio del sistema d'equacions diferencials.
deriv:  funcio que avalua el camp vectorial.

valor retornat: una estimacio de l'error comes en el pas actual.
*/

{
    double t_ponderat,tol1,err,nor,kh,beth,h1;
    int i,j,jj,m;
    if(n>neq){printf("rk45f:error de dimensio(%d and %d)\n",n,neq); exit(1);}
    do {
/* ----->          calculem el valor de les k's.  <-----*/
        m=0;
        for (i=0; i<6; i++)
        {
            t_ponderat=*t+alfa[i]*(*h);
            for (j=0; j<n; j++ ) x_pond[j]=x[j];
            for ( jj=0; jj<i; jj++ )
            {
                ++m;
                beth=*h*beta[m];
                for (j=0; j<n; j++) x_pond[j] += beth*k[jj][j];
            }
            (*deriv)(t_ponderat,x_pond,n,dx);
            for (j=0; j<n; j++ ) k[i][j] = dx[j];
        }
/*-----> Ara calculem els nous punts aproximats per rk4 i rk5 <----*/
        err=0.e0;
        nor=0.e0;
        for (j=0; j<n; j++)
        {
            x4[j]=x[j];
            x5[j]=x[j];
            for (jj=0; jj<6; jj++)
            {
                kh=*h*k[jj][j];
                x4[j] += kh*c4[jj];
                x5[j] += kh*c5[jj];
            }

```

```

        err += fabs(x5[j]-x4[j]);
        nor += fabs(x5[j]);
    }
    err /= n;
/* -----> calculem el nou pas h <-----*/
    tol1=tol*(1+nor/100);
    if (err < tol1) err=MAX(err,tol1/256);
    h1=*h;
    *h*=0.9*pow(tol1/err,0.2e0);
    if (fabs(*h) < hmin ) *h=hmin*SGN(*h);
    if (fabs(*h) > hmax ) *h=hmax*SGN(*h);
} while ((err >= tol1) && (fabs (*h) > hmin));
*t += h1;
for (j=0; j<n; j++) x[j]=x5[j];

return (err);
}

/*-----
*
* Exemple d'integració d'un sistema d'equacions diferencials ordinaries
*
*      x' = x
*      y' = -y
*
* utilitzant el metode de Runge-Kutta-Felberg 4-5 amb control automatic
* de pas.
*
-----*/

#include <stdio.h>
#include <math.h>

FILE *f_res; /*-----> arxiu de resultats <-----*/

/*-----> parametres del RK45 <-----*/

static double hmin=1.e-3; /* Pas maxim que permetem */
static double hmax=1.e-1; /* Pas minim que permetem */
static double h_ini=1.e-2; /* Pas d'integració inicial */
static double tol=1.e-10; /* Error relatiu maxim que permetem */

```

```

static int n=2;          /* Dimensio del sistema */

/*-----> accio que conte les equacions <-----*/

void deriv (double t,double x[], int n, double y[])
{
    y[0] = -x[1];
    y[1] = x[0];
}

void cercle(void)
{
    double t,t_ini,t_final,x[2],r,error;
    double h;
    ini_rk45f (n); //inicialitzem les taules del RK45F
    /*-----> fixem les condicions inicials <-----*/
    t_ini=0.e0;
    t_final=10.e0;
    x[0] = 1.e0;
    x[1] = 0.e0;
    /*-----> calculem la solucio fins un cert temps <-----*/
    t=t_ini;
    h=h_ini;
    while (t<t_final) /*integrem mentre t no sigui mes gran que t_final */
        /* recordem que t_nou= t_act+h */
    {
        error=rk45f(&t,x,&h,tol,hmin,hmax,n,deriv);
        r=x[0]*x[0]+x[1]*x[1]; /* radi de la solucio (nomes per aquest sistema) */
        fprintf(f_res,"t=%11.4e  r=%24.16e  h=%11.5e  er=%11.5e  \n",t,r,h,error);
    }
    h=t_final-t; /* correccio per parar exactament a t_final */
    error=rk45f(&t,x,&h,tol,hmin,hmax,n,deriv);
    r=x[0]*x[0]+x[1]*x[1];
    fprintf(f_res,"t=%11.4e  r=%24.16e  h=%11.5e  er=%11.5e  \n",t,r,h,error);
    /*-----> tanquem fitxer i l'integrador <-----*/
    fclose (f_res);
    fi_rk45f (n); //alliberem les taules del RK45F
}

void main (int argc, char **argv)
{
    char nom_sort[20];
    printf("Nom fitxer de resultats\n");
}

```

```
scanf("%s",nom_sort);
f_res=fopen(nom_sort,"w");
if (f_res==NULL){
    puts("No puc obrir el fitxer");
    exit(0);
}

puts ("tolerancia ?");
scanf("%le",&tol);

    cercle();

}
```