

Llicenciatura de Matemàtiques

FME - UPC

MÈTODES NUMÈRICS I

Pràctiques
Curs 2001–2002

Antoni Guillamon

Antoni Susín

Capítol 0

Llenguatge i eines gràfiques

El llenguatge en què programarem les pràctiques de l'assignatura de Mètodes Numèrics 1 és el llenguatge *C*. Suposarem coneguts una sèrie d'aspectes del llenguatge *C* que ja s'han vist a les assignatures d'Informàtica 1 i 2, com ara els diferents tipus de variables, operadors, estructures de control de flux, etc. Al llarg del curs farem èmfasi en els aspectes que més ens interessin des d'un punt de vista de càlcul numèric.

El compilador que s'utilitzarà aquest any s el MS Visual C++ versió 6.0 dins l'entorn Windows98/NT. Es mirarà de no utilitzar característiques específiques d'aquest compilador per tal d'obtenir programes que es puguin compilar i executar sota diferents sistemes operatius com ara Linux, Unix, etc.

Per la representació gràfica dels resultats utilitzarem la llibreria gràfica OpenGL i GLUT per la gestió de finestres gràfiques. Aquestes eines garanteixen la possible exportació del codi a les altres plataformes ja que també existeixen en els altres sistemes operatius.

0.1 Fitxers i apuntadors

Sense pretendre ser exhaustius, repassarem com es tracten dos dels aspectes que més utilitzarem en contextos numèrics. D'una banda, la lectura i escriptura de dades en fitxers i de l'altra, el tractament de matrius i vectors fent ús de la gestió dinàmica de memòria.

Pel que fa als fitxers, ens cal conèixer les instruccions del llenguatge *C* que serveixen per obrir un fitxer (`fopen`), escriure-hi (`fprintf`) i llegir dades (`fscanf`), així com les seves sintaxis respectives.

Les matrius i vectors en C poden ser tractades com a simples *taules*, però per poder aconseguir una major eficiència pel que fa a les dimensions variables i també a l'emmagatzemament, és preferible tractar-les com a apuntadors en el cas dels vectors, i com a dobles apuntadors en el cas de les matrius. Per això utilitzarem les funcions `calloc` i `malloc` de la llibreria `stdlib.h`. Aquestes funcions ens permeten reservar (en temps d'execució) l'espai de memòria necessari per guardar la matriu o vector amb què estiguem treballant. La diferència essencial entre elles és que `calloc` a més de reservar l'espai de memòria, també inicialitza a zero les components. Com que treballar amb matrius i vectors és un procés que repetirem sovint; el millor és disposar d'unes funcions pròpies que ens facilitin aquesta tasca. Així, dins el fitxer **inici.c** d'exemple que hem preparat, estan incloses les funcions

```
double **crear_matriu (int num_fil_ini, int num_fil_final,
                      int num_col_ini, int num_col_final );

void alliberar_matriu (double **matriu, int num_fil_ini,
                      int num_col_ini, int num_col_final );

double *crear_vector (int num_comp_ini, int num_comp_final );

void alliberar_vector (double *vector, int num_comp_ini );
```

que, com veurem, ens permeten crear i alliberar la memòria necessària per treballar amb matrius i vectors. Aquestes funcions també ens permeten treballar (si volem) amb les components numerades de 1 a n (que en molts casos pot ser més còmode) en comptes de la numeració original del llenguatge C que sempre va de 0 a $n - 1$.

En el fitxer **inici.c** també hi trobareu com es fa per poder introduir els noms dels fitxers de treball per teclat, així com un mètode per donar el temps d'execució d'un programa.

Exercici 0.1.1 *A partir del fitxer **inici.c**, feu un programa que denotarem **ax_b.c** i que ha de verificar que un cert vector x és solució del sistema d'equacions $Ax = b$, on A és una matriu quadrada $n \times n$, i b és el vector de termes independents.*

*Llegireu les dades d'un fitxer que anomenarem **ax_b.dat** i que tindrà a cada línia un únic valor. L'estructura del fitxer serà la que expliquem a continuació. A la primera línia, un valor enter n que correspon a la dimensió. A les $n \times n$ línies següents, els valors de les components de la matriu del sistema A , recorreguda per files. Seguidament, les n components del vector de termes independents b , i també sense cap línia en blanc, les n components del vector de solucions x .*

Una vegada hagueu llegit el fitxer de dades, cal fer la comprovació que el vector x és efectivament la solució. Per això heu de donar per pantalla el residu del problema, que

definirem per

$$r = \| Ax - b \|_{\infty} .$$

(Obs. Es podria utilitzar també qualsevol altra norma vectorial per definir el residu)

0.2 Eines Gràfiques

Quan s'intenta expressar els resultats d'un cert càlcul, generalment el que es fa és escriure aquests per pantalla (o en un fitxer) sota la forma d'un llistat numèric. En molts casos, poder expressar els resultats en forma d'una certa gràfica, o dibuix, ajudaria a una millor comprensió dels mateixos.

En aquesta pràctica, introduïrem de forma simple les eines gràfiques que ens permetran fer dibuixos (2D) a la pantalla i poder representar funcions, etc. Cada compilador, i de fet cada entorn gràfic, té implementades unes funcions de dibuix que **no** són estàndards. Això pot provocar que un cert programa deixi de funcionar quan fem un canvi de compilador o bé, per exemple, que un programa que funcionava en un sistema operatiu no funcioni en un altre.

Per evitar aquests problemes (que de fet no ho són pel que fa al codi en *C* que **sí** és estàndard), el que farem és concentrar les funcions gràfiques en un fitxer que anomenarem `grafics.c`, en el qual *batejarem* les funcions de dibuix més usuals amb nous noms, que són els que utilitzarem en els nostres programes. Així, qualsevol canvi d'entorn ens obligarà a modificar únicament aquest fitxer `grafics.c`, mentre la resta del programa es podrà utilitzar sense cap alteració.

Quan vulguem fer un dibuix a la pantalla haurem de fer essencialment les següents tasques:

1. **Inicialitzar el mode gràfic**, de manera que puguem utilitzar la tarja gràfica. D'això se n'encarregarà la funció

```
void inicia_grafic (char titol[]).
```

on `titol` és el nom que posarem al programa i que sortirà en el requadre superior de la finestra.

2. **Definir finestra de treball**, que és la regió del pla, en coordenades reals, en la qual volem fer el dibuix. Ho farem amb la funció

```
void finestra_dibuix (double XMIN, double XMAX, double YMIN, double YMAX).
```

on (XMIN,YMIN) i (XMAX,YMAX) són els vèrtexs inferior esquerre i superior dret, respectivament, de la regió de dibuix.

3. **Funcions que ens permeten dibuixar.** De manera intuïtiva, el dibuix el farem mitjanant un *cursor* imaginari que anirem movent per sobre de la pantalla. Les accions de dibuix es poden reduir a les següents:

- Moure el cursor a un determinat punt. No implica cap acció de dibuix pròpiament, ja que no es veu res a la pantalla.

```
void mou_a ( double x, double y).
```

- Dibuix d'un punt. El funcionament és com l'anterior, però en aquest cas encendrem el punt corresponent amb el color especificat. El cursor quedarà situat en aquest nou punt.

```
void punt_a (double x, double y, int color).
```

- Dibuix d'una línia recta. Es tracta de dibuixar una línia des de la posició actual del cursor fins a la nova posició.

```
void linia_a (double x, double y, int col).
```

4. **Finalitzar el mode gràfic.** Passa el control a la finestra de text.

```
void fi_dibuix ().
```

0.3 L'aplicació "estàndard"

Amb aquesta pràctica volem obtenir una representació del comportament caòtic d'un sistema dinàmic. Entendrem per sistema dinàmic un cert mecanisme que permet seguir l'evolució (o trajectòria) d'un cert valor inicial. El comportament caòtic d'un sistema es manifesta en la gran diferència entre els valors finals que s'obtenen en variar lleugerament els valors dels paràmetres que poden intervenir en el sistema.

Un exemple d'aquest tipus de sistemes ens el dona l'*aplicació estàndard*, que és una iteració dels punts del pla \mathbb{R}^2 . De fet, és una iteració que podem definir com

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

1. Preneu una finestra de dibuix centrada en el punt (π, π) de 2π de costat. Considereu com a valors inicials els punts situats sobre l'eix $x = \pi$, que recorrerem amb pas $h = \frac{2\pi}{30}$.

2. Fixant un valor $K = 1.11$, per a cada punt inicial pinteu tots els punts de la seva òrbita (els seus iterats) fins a $n = 300$ iteracions. Feu el mateix prenent ara els valors inicials sobre l'eix $y = \pi$.
3. Repetiu el càlcul anterior però per a valors de $K \in (0.8, 3.4)$. Feu una seqüència animada que passi de $K = 0.8$ a $K = 3.4$, amb un pas de 0.2.
4. Repetiu l'animació anterior prenent ara un pas de 0.1 per als valors de K i prenent una finestra de dibuix centrada en el punt (π, π) de 0.5 de costat (farem un zoom de l'anterior).

Feu la pràctica en un arxiu que es digui **standard.c**.

0.4 Iteració gràfica de l'equació logística

Considerem la iteració definida per l'equació:

$$x_{n+1} = \lambda x_n(1 - x_n), \quad x_n, \lambda \in \mathbf{R}, \quad (1)$$

on cal prendre un valor inicial x_0 , i λ és un valor fixat. La **representació gràfica** d'aquesta iteració consisteix en el procés que descriurem a continuació.

Si considerem, en general, una funció d'iteració $x_{n+1} = f(x_n)$, veiem que el valor de les abscisses i les ordenades s'intercanvia a cada pas. Aquest procés es pot representar utilitzant els punts de la bisectriu del primer quadrant, de manera que la representació gràfica d'una iteració consisteix en una poligonal que uneix successivament punts sobre la gràfica de la funció d'iteració i punts sobre la bisectriu. Així, si partim de x_0 (punt $(x_0, 0)$) i calculem el valor de x_1 (punt $(x_0, f(x_0))$), farem la recta que uneixi aquests dos punts i la continuarem de manera horitzontal fins a tallar la bisectriu (punt (x_1, x_1)). Després calcularem el següent iterat $x_2 = f(x_1)$ i unirem el punt anterior de la bisectriu amb aquest nou punt (x_1, x_2) , i repetirem el procés un nombre finit N (prou gran) de vegades.

Normalment el que es pretén estudiar és el comportament límit de la successió $(x_n)_{n \in \mathbf{N}}$, i veure com depèn aquest dels diferents paràmetres de la iteració. En el nostre cas estudiarem el comportament en funció del valor inicial x_0 i de λ .

Exercici 0.4.1 *A partir de l'exemple de representació de la funció $f(x) = \sin x$, feu la representació dels eixos, la bisectriu i la funció d'iteració logística 1 per $\lambda = 2.8$, dins la finestra d'extrems $(-0.3, -0.3)$ i $(1.3, 1.0)$.*

Exercici 0.4.2 Per $N = 200$ i $\lambda = 2.8$ representeu les iteracions per diferents punts inicials x_0 . Feu que el programa demani x_0 per teclat i experimenteu amb diferents valors, per exemple 0.1, 0.2, 0.4, 0.6, 0.8, 0.9 (**Obs.** Què passaria si prenguéssim $x_0 = 0$?).

Posteriorment feu que també demani per teclat el nombre màxim de valors d'iteracions N i el valor del paràmetre λ . Vegeu què passa quan donem diferents valors a λ , per exemple 3.1, 3.4, 3.7, 3.9.

Exercici 0.4.3 Feu el diagrama de bifurcacions de Feigenbaum, que s'obté al variar els diferents valors de λ i representar només els valors límit de les successions d'iteració.

Fixeu $N = 200$, finestra de dibuix amb eixos (x, λ) i valors $(-0.3, 2.8)$ i $(1.3, 4)$ respectivament per als seus extrems. Prenem sempre un punt fixat $x_0 = 0.9$, i per a cada valor de λ iterem aquest punt inicial. Donat que ens interessa el comportament en el límit, podem representar només els punts (x_n, λ) a partir de $n = 50$ (per evitar el soroll inicial). Aquest procés el repetirem des de $\lambda_{min} = 2.8$ fins a $\lambda_{max} = 4$ fent 300 subdivisions.

Capítol 1

Errors

En aquest capítol, a través de pràctiques breus, volem mostrar i donar solucions a alguns dels problemes d'errors més generals que sorgeixen en el tractament numèric: com sumar una sèrie? com trobar una bona aproximació racional d'un nombre real? com atacar els algorismes recurrents? com evitar cancel·lacions?...

1.1 Sumació de la sèrie harmònica

La sumació de sèries ofereix certes dificultats que cal tenir en compte, i que il·lustrarem en aquesta pràctica. A més, s'aprofitarà per introduir el concepte de *fracció contínua*, emprat en l'aproximació de nombres reals per racionals.

La sumació de la sèrie harmònica

$$\sum_{n=1}^{+\infty} \frac{1}{n^\alpha}$$

va ser un dels problemes que va treure la son a L. Euler. No és gens difícil amb el que coneixem avui en dia decidir la convergència de la sèrie quan $\alpha \in \mathbb{R}$. En canvi, donar la suma exacta de la sèrie en els casos en què aquesta convergeix és un problema encara obert. Euler va trobar una manera ben original (d'entrada no sembla res més que un joc formal) de calcular-la per a $\alpha = 2r$, amb $r \in \mathbb{N}$, que reproduïm a l'apèndix enquadrat al final de la secció.

Malgrat tot, però, no s'ha trobat encara cap idea que permeti atacar el cas senar. Una conjectura plausible però poc profunda seria afirmar, a la vista dels resultats per al cas

parell, que

$$\sum_{n=1}^{+\infty} \frac{1}{n^r} = \beta(r)\pi^r, \text{ amb } \beta(r) \in \mathbb{Q}, r \text{ senar.}$$

L'objectiu d'aquesta pràctica és trobar aproximacions numèriques de $\beta(r)$.

Exercici 1.1.1 *Construïu un programa en C que treballi amb precisió simple (amb la intenció de detectar més ràpidament el fenomen que volem mostrar), seguint les següents pautes:*

i) Creeu el fitxer **euler.c**, amb la següent inicialització (com a mínim):

```
int r;
long int M;
float s1, s2;
float pi=3.1415927;
FILE *sortida;
sortida= fopen("euler.sor","w");
```

Observeu que preparem la variable **M** perquè sigui de tipus **long int**, ja que preveiem sumes fins a índexs alts.

ii) Per a cada r , definim les sumes parcials $S1_m := \sum_{n=1}^m \frac{1}{n^r}$ i $S2_m := \sum_{n=m}^1 \frac{1}{n^r}$. El programa ha de calcular $S1_m$ fins a un M que compleixi que $|(S1_{M+1} - S1_M)/S1_M| < 10^{-8}$. Definim aleshores

$$s1 := \frac{S1_{2M}}{\pi^r}, \quad s2 := \frac{S2_{2M}}{\pi^r}.$$

- iii) Feu córrer el programa per a $r = 2$. Per quin valor us decantaríeu, s_1 o s_2 ? Justifiqueu-ho (utilitzeu la informació de l'Apèndix del requadre).
- iv) Escriviu en el fitxer **euler.sor** una taula que a cada fila contingui r , M , **s1** i **s2** per a $r \in \{2, 3, 5, 7, 9, 11, 13\}$.

Apèndix: Suma efectiva d'una sèrie harmònica.

La funció $\frac{\sin x}{x}$ té zeros en els punts de la forma $x = n\pi$, amb $n \in \mathbb{Z} \setminus \{0\}$.
D'altra banda, el seu desenvolupament formal en sèrie de Taylor és:

$$\frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots$$

Si aquest desenvolupament el tractem com un "polinomi de grau ∞ ", s'ha de complir que

$$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots = \prod_{n=1}^{+\infty} \left(1 - \frac{x^2}{n^2\pi^2}\right)$$

Igualant potències en aquesta darrera expressió, s'obté, per exemple amb les x^2 , que:

$$\frac{1}{3!} = \sum_{n=1}^{+\infty} \frac{1}{n^2\pi^2},$$

d'on es conclou que $\sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. Amb una mica més de paciència, en teoria es poden anar obtenint les sumes per a $\alpha = 4, 6, 8, \dots$

1.2 La fracció contínua d'un nombre real

Buscarem ara l'expressió en forma de fracció del nombre $\beta(r)$, introduït a la Pràctica 1.1. La millor aproximació racional d'un nombre real es troba buscant-ne la seva **fracció contínua**, que es pot pensar simbòlicament com una aplicació que a cada nombre real li fa correspondre una successió de nombres naturals seguint el següent algorisme:

Donat un nombre real x , es calculen les successions recurrents $(a_n)_n$ i $(b_n)_n$ donades per:

$$a_0 = [x]; \quad b_0 = x - a_0; \quad a_n = \left[\frac{1}{b_{n-1}}\right]; \quad b_n = \frac{1}{b_{n-1}} - a_n,$$

on $[y]$ denota la part entera d'un nombre real y .

Aleshores, $x(m)$, el nombre racional obtingut al confegir els primers m termes de la

successió $(a_n)_n$ segons la fórmula

$$x(m) = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ddots a_m}}},$$

és una aproximació racional del nombre real x (la fracció contínua de x d'ordre m .)

Exercici 1.2.1 Modificarem la pràctica proposada a l'Exercici 1.1.1 amb l'objectiu de calcular la fracció contínua de $\beta(r)$ (prenent l'aproximació donada per s_2).

- i) Copieu el fitxer **euler.c** en un fitxer anomenat **fraccont.c**, i obriu-lo.
- ii) Creeu una acció que calculi la fracció contínua d'un nombre **s**, i que sigui del següent tipus:

```
long int *fraccont(double s,int nfc).
```

El resultat, doncs, serà un apuntador a un vector en el qual emmagatzemarem els termes de la successió (a_n) . El paràmetre **nfc** conté el nombre màxim de termes de la fracció contínua (és a dir, que calculem a_0, \dots, a_{nfc}).

- iii) Per a cada $r \in \{2, 3, 5, 7, 9, 11, 13\}$, calculeu la fracció contínua d'ordre 100 de **s2** (aneu amb compte perquè **s2** era originàriament una variable **float**!)
- iv) Guardeu al fitxer **fraccont.sor** la successió a_0, \dots, a_{100} , per a cada valor de r .

1.3 Minimitzant la propagació de l'error

El present exemple és un *clàssic* quant a il·lustració de l'ordre en què s'han d'efectuar certs processos. La filosofia és similar a la de la Pràctica 1.1.

Exercici 1.3.1 Considereu la integral definida

$$E_n = \int_0^1 x^n e^{x-1} dx. \tag{1.1}$$

- i) Demostreu que $E_n = 1 - nE_{n-1}$.

- ii) Partint d' E_0 , obteniu de manera recurrent un valor aproximat d' E_{12} , treballant amb precisió simple. En general, a les aproximacions obtingudes així, les bategem com ϵ_n .
- iii) Observeu que el valor de E_{12} és negatiu? Si no és així, repasseu el que heu fet. Si és així, decidiu si és possible, observant la definició (1.1), i doneu una explicació del fet.
- iv) Corregiu el defecte anterior emprant que $E_{20} \simeq 0$ i calculeu a partir d'aqu un nou valor aproximat d' E_{12} . Els valors obtinguts d'aquesta forma els denotarem per ϵ'_n .
- v) Construïu al fitxer `iterint.sor` una taula a 4 columnes, amb les següents entrades: n , ϵ_n , ϵ'_n , E_n . Per a aquesta darrera columna, trobareu els valors d' E_n en precisió simple, al fitxer `iterint.c`, a la variable `dv`.

1.4 Sumació d'una sèrie amb signes alternats

Suposeu que volem calcular e^x , essent x un nombre negatiu. Us proposem un experiment, amb dues alternatives, per exhibir els errors numèrics que s'hi poden produir.

Per al propòsit d'aquesta prctica, emprarem el desenvolupament de Taylor, $P_N(x, a)$, d'ordre N de la funció e^x al voltant d' a . El fenomen que posarem de manifest no es dona noms amb els polinomis de Taylor, ni noms amb la funció exponencial, sinó amb qualsevol suma de signes alternats.

Exercici 1.4.1 *Elaboreu un fitxer `taylor.c` on es calculi el polinomi de Taylor d'ordre N de la funció $f(x) = e^x$ al voltant d' $a = 0$, i es donguin els resultats per omplir la Taula 1.1. Prepareu-lo perquè els valors N i x que necessiteu s'entrin a partir d'un fitxer `taylor.dad`.*

A cadascun dels 24 quadres definits a la Taula 1.1 hi heu d'introduir els 3 valors següents:

- a) e^x emprant directament l'acció construïda;
- b) e^x emprant l'acció construïda, però aquesta vegada aplicada a e^{-x} ;
- c) e^x usant la funció `exp` de la llibreria `math.h`.

Comenteu el perquè dels aspectes numèrics que destaquin dels resultats obtinguts:

- quina de les dues opcions ((a) o (b)) s'és més fiable?

	$N = 5$	$N = 10$	$N = 20$	$N = 50$
Precisió simple $x = -2$ $x = -6$ $x = -8$				
Doble precisió $x = -2$ $x = -6$ $x = -8$				

Taula 1.1:

- *quins errors són deguts al mètode i quins a la precisió?*
- *quines diferències es noten entre la precisió simple i la precisió doble?*
- ...

1.5 Solució numèrica d'una equació de segon grau

És ben conegut que les solucions de l'equació $ax^2 + bx + c = 0$, amb $a \neq 0$, són

$$x_{+,-} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Malgrat tot, des del punt de vista numèric no sempre va bé emprar aquesta fórmula.

Exercici 1.5.1

i) *Observeu què passa si l'empreu amb l'equació*

$$x^2 - 100000x + 0.0000001,$$

suposant que treballeu amb precisió simple (en general, proveu-ho per a equacions tals que $b \gg c$).

ii) *Se us acut alguna alternativa de càlcul? Utilitzeu-la per donar unes solucions més correctes de l'equació anterior.*

Indicació: *Podeu trobar una pista en les dues frases següents: Molta unitat la té i pot lluir-la ignominiosament qui uneix europeus. Ulisses pot estendre llit còmode on no jau un gat arraulit tremolós.*

Capítol 2

Interpolació

2.1 Interpolació. Fenomen de Runge

En aquesta secció pretenem il·lustrar el que es coneix com a Fenomen de Runge. Es tracta d'estudiar el comportament d'un polinomi d'interpolació de grau suficientment gran. Per això, el que farem és comparar la representació d'una funció amb la del seu polinomi interpolador. El polinomi el construirem de dues formes: en primer lloc, utilitzant una xarxa de punts equiespaiats, i posteriorment utilitzant les abscisses de Txeixev.

2.1.1 Interpolació de Newton. Diferències dividides

Primerament, caldrà construir el polinomi d'interpolació d'una certa funció $f(x)$ en una xarxa de punts $x_i \in [a, b], i = 0 \div n$. Aquest polinomi serà de grau n i el denotarem per $P_n(x)$. Atès que aquest polinomi és únic, qualsevol dels mètodes de construcció ens donarà el mateix resultat. Nosaltres implementarem el mètode de Newton o de diferències dividides.

Exercici 2.1.1 *Programeu una acció per a calcular el polinomi d'interpolació $P_n(x)$ a partir d'una taula de valors corresponents als punts d'interpolació x_i i les seves imatges $f_i = f(x_i), i = 0 \div n$. L'acció s'anomenarà `dif_div`, serà de tipus `void` i tindrà la següent passa de paràmetres:*

```
void dif_div (double x[], double f[], double p[], int n),
```

on

- $x[]$ és un vector $(n + 1)$ -dimensional que conté la xarxa de punts en els quals interpolem.
- $f[]$ és un vector $(n + 1)$ -dimensional que conté les imatges dels punts anteriors per la funció $f(x)$ que volem interpolar.
- $p[]$ és un vector $(n + 1)$ -dimensional que conté els coeficients del polinomi interpolador expressat a partir de la forma de Newton, és a dir, són els primers valors de cada una de les columnes de la taula de diferències dividides.
- n és el grau del polinomi interpolador. Recordeu que la numeració es fa (de forma anàloga al llenguatge C) des de 0 a n , i per tant, el problema és $(n + 1)$ -dimensional.

Exercici 2.1.2 Programeu també una funció per a calcular el valor que pren un cert polinomi interpolador en un punt donat. En aquest cas caldrà tenir present que el polinomi està expressat de la forma que retorna l'acció `dif_div` programada anteriorment. La funció de substitució l'anomenarem `pol_subs`, serà de tipus `double` i retorna el valor que pren el polinomi en el punt. La passa de paràmetres ha de ser:

```
double pol_subs (double c, double x[], double p[], int n),
```

on

- c és el punt que volem substituir.
- $x[]$ és un vector $(n + 1)$ -dimensional que conté la xarxa de punts en els quals interpolem.
- $p[]$ és un vector $(n + 1)$ -dimensional que conté els coeficients del polinomi interpolador obtinguts per l'acció `dif_div`.
- n és el grau del polinomi interpolador.

2.1.2 Fenomen de Runge

El Fenomen de Runge es manifesta quan utilitzem polinomis de grau gran ($n \geq 10$) per interpolar abscisses equiespaiades. La funció interpolada i el seu polinomi interpolador poden discrepar notablement en els valors dels punts que no són de la xarxa, per la qual cosa el polinomi interpolador no seria una bona aproximació de la funció. Aquesta discrepància és més manifesta en els extrems de l'interval que en el seu centre. Com veurem, es millora

el resultat si es prenen com a punts d'interpolació les abscisses de Txeixev, tot i que si el nombre de punts a interpolar és prou gran, sempre serà més aconsellable treballar amb interpolacions parcials de grau baix, tipus *spline*.

Per il·lustrar aquest fenomen, considerarem la funció $f(x)$ dins l'interval $[a, b]$ i prendrem dues xarxes de punts, la primera equiespaiada

$$x_i = a + i \frac{b-a}{n}, \quad i = 0 \div n,$$

i la segona formada per les abscisses de Txeixev que tenen per expressió

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos \left(\frac{(2i+1)\pi}{2(n+1)} \right) \quad i = 0 \div n.$$

Exercici 2.1.3 *i) A partir de les accions programades en la primera part, feu un programa que dibuixi els eixos de coordenades i les tres gràfiques superposades corresponents a la funció*

$$f(x) = \frac{1}{1+25x^2},$$

i als seus polinomis interpoladors per a les dues xarxes de punts esmentades anteriorment. Preneu com a interval d'interpolació $[-1, 1]$ i feu que el grau d'interpolació es demani per teclat.

ii) Calculeu també, de manera aproximada, el valor de la diferència màxima entre la funció i els polinomis d'interpolació respectius. Doneu aquest resultat per pantalla.

Capítol 3

Aplicacions de la interpolació

3.1 Zeros de funcions

Volem calcular els zeros d'una certa funció $f(x)$ dins un interval $[a, b]$. El procediment que seguirem és, en primer lloc, separar els possibles zeros (considerant que la funció sigui contínua) fent un escombrat de l'interval d'estudi i detectant possibles canvis de signe. Una vegada determinat el subinterval en el qual hi ha un zero, aplicarem algun dels mètodes de càlcul de zeros. En aquesta pràctica, el que farem és calcular per duplicat els zeros, per poder fer un estudi comparatiu de les diferents velocitats de convergència dels mètodes de Newton i de la secant.

Exercici 3.1.1 *Considerem la funció $f(x) = 2x - \ln x - 4$ dins de l'interval $[0, 4]$. Volem conèixer els seus zeros amb una precisió $\text{tol} < 1.e-12$.*

- *Dibuixeu la gràfica de la funció $f(x)$.*
- *Calculeu els seus zeros pels mètodes de Newton i la secant. Si el canvi de signe correspon al subinterval $[a_k, b_k]$, preneu aquests valors com a inicials en el cas de la secant i preneu a_k com a valor inicial en el cas de Newton.*

Doneu els resultats en un fitxer de resultats que anomenarem `zeros.out` cada una de les iteracions necessàries per a calcular el zero corresponent amb els dos mètodes. Per uniformitzar, a cada línia podeu escriure tres valors: un valor `int` corresponent a la iteració, un `double` que sigui l'aproximació actual del zero que volem calcular, i finalment un altre `double` que és el valor que pren $f(x)$ en aquest punt.

3.2 Sensibilitat del mètode de Newton per a zeros de funcions

El mètode de Newton per a zeros de funcions es pot pensar com un *sistema dinàmic discret* que a cada punt x li assigna la seva òrbita. Així, si el mètode convergeix, l'òrbita tendirà a un únic punt. Vist des d'aquesta òptica, donada una solució, x^* , de l'equació $f(x) = 0$, direm que un punt x_0 està a la *conca d'atracció* de x^* si la successió recurrent definida per

$$x_{n+1} = x_n - f(x_n)/f'(x_n), \quad n \geq 0; \quad (3.1)$$

tendeix a x^* .

Si la convergència a un únic punt no es dona, el sistema dinàmic discret pot presentar diferents comportaments. Per exemple, si existeixen nombres naturals N i p tals que $x_n = x_{n+p}$, per a tota $n \geq N$, direm que l'òrbita tendeix a l'*òrbita periòdica* de període p formada per $\{x_N, x_{N+1}, \dots, x_{N+p-1}\}$.

Aquesta mena de comportaments són els que intentem evidenciar i analitzar en aquesta pràctica; a la primera part, els polinomis seran de variable real i a la segona, de variable complexa.

En tota la pràctica, denotarem

$$P_\alpha(x) := x^3 + \alpha x + 1,$$

i cada alumne tindrà dues α pròpies, α_1 i α_2 , que es calcularan de la següent manera:

A cada lletra del vostre cognom, li assigneu un valor numèric segons la taula, sumeu tots els valors i dividiu per $12.5N$, on N s la longitud del cognom. Digueu-li A al número obtingut. Llavors, $\alpha_1 = A/600 - 1.273$, $\alpha_2 = A/5 - 0.2$.

A	B	C	D	E	F	G	H	I	J	K	L	\dots	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	\dots	20	21	22	23	24	25

Per exemple, si algú es diu Newton, calcularà les seves alfes fent:

$$\begin{aligned} A &= \frac{13+4+22+19+14+13}{75} = \frac{17}{15} \approx 1.1333333, \\ \alpha_1 &= 17/9000 - 1.273 \approx -1.2711111, \\ \alpha_2 &= 17/75 - 0.2 \approx -0.026666667. \end{aligned}$$

Tal com passaria en el cas d'en Newton, si la part decimal és infinita, cal prendre una aproximació.

3.2.1 El mètode de Newton per a polinomis de variable real

La representació gràfica de la iteració (3.1) consisteix en el procés que descriurem a continuació.

Si considerem, en general, una funció d'iteració $x_{n+1} = f(x_n)$, veiem que el valor de les abscisses i les ordenades s'intercanvia a cada pas. Aquest procés es pot representar utilitzant els punts de la bisectriu del primer quadrant, de manera que la representació gràfica d'una iteració consisteix en una poligonal que uneix successivament punts sobre la gràfica de la funció d'iteració i punts sobre la bisectriu. Així, si partim de x_0 (punt $(x_0, 0)$) i calculem el valor de x_1 (punt $(x_0, f(x_0))$), farem la recta que uneixi aquests dos punts i la continuarem de manera horitzontal fins a tallar la bisectriu (punt (x_1, x_1)). Després calcularem el següent iterat $x_2 = f(x_1)$ i unirem el punt anterior de la bisectriu amb aquest nou punt (x_1, x_2) , i repetirem el procés un nombre finit N (prou gran) de vegades.

Normalment el que es pretén estudiar és el comportament límit de la successió $(x_n)_{n \in \mathbb{N}}$, i veure com depèn aquest dels diferents paràmetres de la iteració. En el nostre cas estudiarem el comportament en funció del valor inicial x_0 i de α .

Exercici 3.2.1 Feu la representació dels eixos, la bisectriu i la funció d'iteració

$$x - P_\alpha(x)/P'_\alpha(x),$$

per a $\alpha = \alpha_1$, dins la finestra d'extrems $(-2, -2)$ i $(2, 2)$.

Exercici 3.2.2 Per a $N = 200$ i $\alpha = \alpha_1$, representeu les iteracions d'un punt inicial x_0 . Feu que el programa demani x_0 per teclat i experimenteu amb diferents valors.

Busqueu a partir de la gràfica, alguna òrbita periòdica de la iteració, amb període > 1 . Un cop localitzada, modifiqueu el programa perquè la detecti i guardeu en un fitxer **orbper.out** els punts de l'òrbita.

Posteriorment feu que també demani per teclat el nombre màxim d'iteracions, N , i el valor del paràmetre α . Vegeu què passa quan donem diferents valors a α ; experimenteu bàsicament entre -2 i -1 .

NOTA: En aquest exercici hi ha part d'experimentació. El programa que heu d'entregar ha de tenir ja implementada la part de recerca dels punts de l'òrbita periòdica i mostrar-la per pantalla, amb el punt inicial adient. És a dir, heu d'entregar una òrbita periòdica (el seu dibuix i el llistat de punts).

Exercici 3.2.3 Feu el diagrama de bifurcacions de Feigenbaum, que s'obté al variar els diferents valors d' α i representar només els valors límit de les successions d'iteració.

Fixeu $N = 300$, finestra de dibuix amb eixos (x, α) i valors $(-2, -2)$ i $(2, -1)$ respectivament per als seus extrems. Prenem sempre un punt fixat $x_0 = 0.$, i per a cada valor d' α iterem aquest punt inicial. Donat que ens interessa el comportament en el límit, podem representar només els punts (x_n, α) a partir de $n = 250$ (per evitar el soroll inicial). Aquest procés el repetirem des de $\alpha_{\min} = -2$ fins a $\alpha_{\max} = -1$ fent 1000 subdivisions.

Exercici 3.2.4 Feu un zoom del diagrama de bifurcacions de l'apartat anterior al domini $D = \{(x, \alpha) : -0.19 \leq x \leq 0.22; -1.3 \leq \alpha \leq -1.25\}$.

Exercici 3.2.5 Estructureu tota la pràctica fins a aquest punt en un menú inicial que deixi triar entre: a) Representació i llistat de l'òrbita periòdica (Exercicis 3.2.1 i 3.2.2); b) Diagrama de bifurcacions de l'Exercici 3.2.3; c) Zoom de l'Exercici 3.2.4.

3.2.2 El mètode de Newton per a polinomis de variable complexa

Exercici 3.2.6 Calculeu les arrels w_i de P_{α_2} . Denoteu per w_1 l'única arrel real del polinomi. Definiu $\text{cx} = (w_1 + \text{Re}(w_2))/2$, $\text{fx} = |w_1 - \text{Re}(w_2)|$ i $\text{fy} = 2|\text{Im}(w_2)|$.

Sigui Ω la regió de \mathbb{C} donada per

$$\Omega = \{x + iy : \text{cx} - \text{fx} \leq x \leq \text{cx} + \text{fx}, -\text{fy} \leq y \leq \text{fy}, i = \sqrt{-1}\}.$$

Exercici 3.2.7 Per a cada z_0 que pertanyi a Ω (o dit d'una altra manera, per a cada píxel), considereu la successió $\sigma(z_0)$ de punts de \mathbb{C} obtinguts en les iteracions del mètode de Newton per a $P_{\alpha_2}(z)$.

Segons el valor inicial z_0 que prengueu, poden donar-se diverses situacions:

- a) que la successió recurrent generada s'acosti a w_1 ;
- b) que s'acosti a w_2 ;
- c) que s'acosti a w_3 ;
- d) que se'n vagi de la regió Ω i s'escapi a infinit;

e) que oscil·li entre diversos valors, dins o fora de Ω .

L'exercici consisteix en assignar a cada píxel z_0 un color (n'utilitzarem cinc – $C1, \dots, C5$ – però se'n podria ampliar el nombre), segons el comportament de la successió recurrent associada. Seguirem el següent criteri:

Per a cada z_0 , aneu generant termes de la successió $\sigma(z_0)$. Si al cap de m_1 iteracions ($m_1 = 100$, per exemple), o abans, es compleix que $|z_{m_1} - w_j| < 10^{-3}$, per a alguna $j = 1, 2, 3$, pinteu el píxel associat a z_0 del color C_j . Si, per altra banda, la successió ha sortit de Ω , pinteu el píxel corresponent amb el color C_4 .

Si encara no podeu decidir el color, pinteu-lo d'un color $C5$ (“color de la indecisió”) i acabeu l'experiment.

Un consell de caire estètic: preneu $C1 = \text{vermell}$, $C2 = \text{verd}$, $C3 = \text{groc}$, $C4 = \text{negre}$, $C5 = \text{marró}$.

Al final, s'obté una distribució colorimètrica de la pantalla que ens indica la sensibilitat als valors inicials del Mètode de Newton i ens deixa *retratades* les conques d'atracció dels tres zeros del polinomi $P_{\alpha_2}(z)$. Aquesta distribució és una **imatge fractal**, de la qual la propietat visualment més interessant s' *autosimilaritat*.

Exercici 3.2.8 *L'autosimilaritat es manifesta a les fronteres de les conques d'atracció. Trieu una finestra 0.1×0.1 situada damunt d'una d'aquestes fronteres, i amplieu l'experiment de l'Exercici 3.2.7 amb els punts z_0 pertanyents a aquesta nova finestra. Observeu que l'única cosa que fem s canviar els punts inicials, però no pas les arrels del polinomi.*

Guardeu el fitxer sota el nom **newfrac.c**.

3.3 Integrals abelianes

En l'estudi de bifurcacions d'algunes famílies d'equacions diferencials al pla, s' necessita el còmput d'integrals de funcions $f(x, y)$ sobre les corbes de nivell d'una determinada funció $H(x, y)$. Aquestes integrals solen rebre el nom d'*integrals abelianes*, en honor al matemàtic noruec N.H. Abel. Algunes d'elles tenen relació també amb altres conceptes dinàmics o geomètrics associats a les corbes de nivell. Així, per exemple, l'àrea encerclada per una corba de nivell tancada ve donada, pel Teorema de Green, per

$$A(c) = \int_{H(x,y)=c} x \, dy.$$

Es pot demostrar també que la derivada d' $A(c)$,

$$T(c) = \int_{H(x,y)=c} \frac{dy}{H_x(x,y)},$$

s'el període de la corba $H(x,y) = c$ pensada com la solució d'un cert sistema d'equacions diferencials ordinàries associat a $H(x,y)$.

Sovint, l'expressió de $H(x,y)$ no permet allar-hi una variable respecte l'altra. A més, si les corbes de nivell de $H(x,y)$ són tancades, aleshores s'imponeix donar una funció explícita que ens les defineixi.

Amb tot això, doncs, la resolució d'aquestes integrals s'notablement difícil i per tant, hom recorre sovint a càlculs numèrics per tenir una orientació sobre el problema.

En aquesta pràctica, us proposem calcular les funcions àrea i període per a la funció $H(x,y) = (e^x - x) + (e^y - y)$.

Exercici 3.3.1 Prospeccions analítiques.

Primer de tot, cal que ens fem una idea de la forma de les corbes de nivell de la funció $H(x,y)$.

1. Demostreu que cada corba de nivell $H(x,y) = c$, amb $c \geq 2$, talla dues vegades cada un dels eixos $x = 0$ i $y = 0$. Anomenem $Y_1 \leq 0 \leq Y_2$ i $X_1 \leq 0 \leq X_2$ les ordenades i les abscisses, respectivament, d'aquests punts de tall. Observeu aquests punts i la forma de la corba de nivell a la Figura 3.1.
2. Proveu també que per a cada $x_0 \in [X_1, X_2]$ hi ha dos nics talls $((x_0, y'_0), (x_0, y''_0))$ de $x = x_0$ amb la corba de nivell c i que $y'_0 \leq 0 \leq y''_0$. El mateix passa per a cada $y_0 \in [Y_1, Y_2]$.

Exercici 3.3.2 Realització numèrica.

- A. Supposeu que tenim fixat un valor de $c \geq 2$ i volem efectuar una de les integrals abelianes apuntades més amunt. Ens fixem una tolerància $10^{-\text{tol}}$, fent que `tol` s'entri per teclat.
 - (a) Programeu una acció que dugui a terme el Mètode de Newton per a trobar solucions y de l'equació $e^y - y = c - e^x + x$, essent x un valor fixat. L'acció i la passa de paràmetres han de ser del següent tipus:

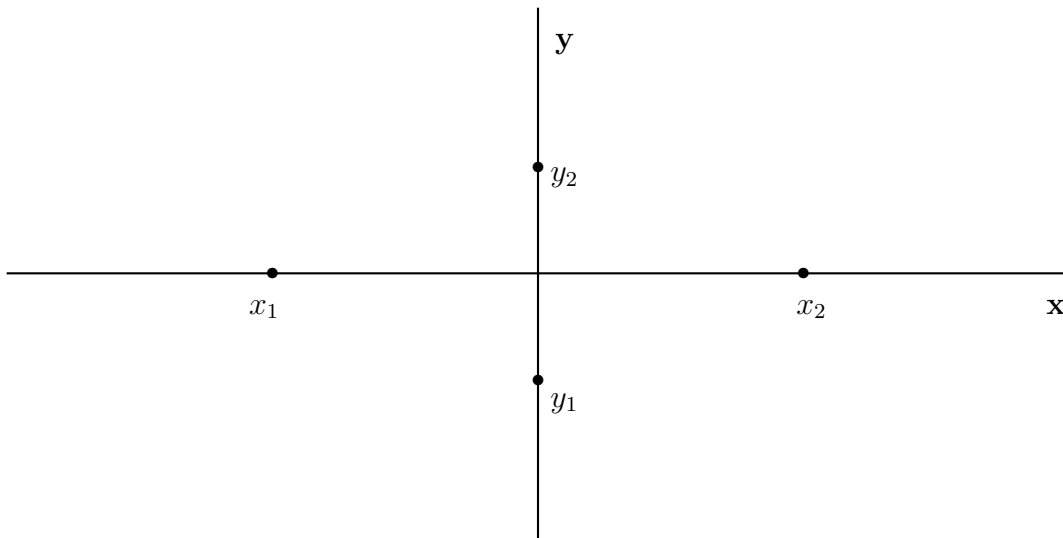


Figure 3.1: Una corba de nivell $H(x, y) = c$ amb els límits indicats.

```
double sirisaac(int tol,double c,double x0,int signe),
```

on x_0 s el punt inicial de la iteració (preneu sempre $x_0 = \text{signe} \sqrt{2(c - e^x + x)}$) i tol és una mesura de la precisió. El procés s'aturarà quan $|f(x_n)| < 10^{-\text{tol}}$. El valor enter signe serveix per indicar si cerquem la solució positiva ($\text{signe} = 1$) o negativa ($\text{signe} = -1$).

(b) Donat un valor de c , utilitzeu l'acció `sirisaac` per trobar Y_1 i Y_2 .

(c) Recordeu que per a cada $y \in (Y_1, Y_2)$, existeixen dos valors de x , $x_1(y) < 0 < x_2(y)$, tals que $(x_i(y), y)$ pertanyen a la corba $H(x, y) = c$. Prepareu una acció que divideixi l'interval $[Y_1, Y_2]$ en n subinterval de longitud h i que per a cada $y_j = Y_1 + jh$, amb $j = 0 \div n$, calculi els punts $x_i(y_j)$, per a $i = 1 \div 2$. L'acció i la passa de paràmetres han de ser del següent tipus:

```
void punts( int tol, double c, double Y1,double Y2,
            int n,double *y,double *x1,double *x2)
```

A l'apuntador `y` cal col·locar-hi els $n + 1$ valors y_j , mentre que els elements dels apuntadors `x1` i `x2` es defineixen a partir de $x1[j] = x_1(y_j)$ i $x2[j] = x_2(y_j)$, amb $j = 0 \div n$. Per tant, aquests dos apuntadors també tindran longitud $n + 1$.

Observeu que, fixada y_j , $x_i(y_j)$ són les solucions de l'equació

$$e^x - x = c - e^{y_j} + y_j;$$

així, doncs, en aquest pas haureu de tornar a utilitzar l'acció `sirisaac` per a calcular-les.

- (d) Programeu una acció que dugui a terme el Mètode de Simpson per al càlcul d'integrals, $I(n)$, donat un apuntador `f` que contindrà els valors de f que cal usar com a imatges a la integració. L'acció i la passa de paràmetres han de ser del següent tipus:

```
double simpson(double *f,double h,int n),
```

on `h` s el pas d'integració i `n` el nombre d'interval·ls.

- (e) Per a cada integral $I(n)$ que calculem, prendrem $n = 2^k$, amb $k = 1, 2, \dots$ fins que

$$\frac{|I(n) - I(n/2)|}{|I(n)|} < 10^{-\text{tol}}.$$

Penseu la millor manera de no repetir avaluacions de les funcions.

- (f) Emprant les accions anteriors, programeu una acció `intabe` que calculi $A(c)$ i $T(c)$, tenint en compte que

$$\int_{H(x,y)=c} f(x,y) dy = \int_{Y_1}^{Y_2} f(x_1(y), y) dy - \int_{Y_1}^{Y_2} f(x_2(y), y) dy.$$

L'acció i la passa de paràmetres han de ser del tipus:

```
double intabe( int tol, double c,double Y1,double Y2,int n,
              double *y,double *x1,double *x2,double (*fun)(double)),
```

on `*fun` s la funció a integrar, que podeu haver definit a la capalera del programa.

- B. Preneu ara, a $c \in [3, 5]$, 201 valors equiespaiats de c i amb tot el programat al punt (A).

- (a) Doneu uns vectors `A` i `T` que donguin les aproximacions numèriques obtingudes, respectivament, per a $A(c)$ i $T(c)$, amb $c \in [3, 5]$.
- (b) Mitjanant diferències centrades, calculeu un vector `A1` que contingui les derivades numèriques d' $A(c)$, obtingudes a partir de les dades d'`A`.
- (c) Representeu simultàniament les gràfiques d' $A(c)$, $T(c)$ i $A'(c)$ emprant les dades recollides als dos punts anteriors. Fixeu-vos que, teòricament, $T(c) = A'(c)$ i per tant, que els punts de `T` i de `A1` han de ser molt similars.

En tota la pràctica, porteu un control del temps d'execució i mostreu-lo al final.

3.4 La taula de la distribució $N(0, 1)$

L'objectiu d'aquesta pràctica es el de dibuixar i tabular la funció de distribució d' X , la variable aleatòria normal amb esperança 0 i variància 1 ($N(0, 1)$).

3.4.1 Introducció

La funció de densitat de probabilitat d' X és:

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2},$$

i la funció de distribució d' X és:

$$F_X(x) = \int_{-\infty}^x f_X(s) ds.$$

Òbviament, com que $f_X(x)$ s la f.d.p. d'una variable aleatòria tenim que

$$\lim_{x \rightarrow \infty} F_X(x) = \int_{-\infty}^{\infty} f_X(s) ds = 1.$$

Com és ben conegut, $F_X(x)$ no admet primitiva en termes de funcions elementals. Per tant, la única manera de calcular $F_X(x)$ per valor concrets de x es la integració numèrica de $f_X(x)$. Per ajudar a aquesta tasca notem que $F_X(-x) = 1 - F_X(x)$ ja que $f_X(x)$ és una funció parella ($f_X(x) = f_X(-x)$). En particular $F_X(0) = 1/2$.

Exercici 3.4.1 *A fi de dibuixar i tabular la funció F_X programeu un procediment*

```
double simpson_normal(double x, double h, double eps)
```

que doni com a resultat el valor de

$$F_X(x+h) - F_X(x) = \int_x^{x+h} f_X(s) ds$$

*calculat pel Mètode de Simpson, amb error absolut inferior a **eps**. Aquest procediment ha ser vàlid per a qualsevol x , h i **eps** tals que $0 \leq x < x+h \leq 4$. Òbviament, abans de programar aquest procediment, cal fer un estudi analític en funció de x , h i **eps** del pas que cal utilitzar al aplicar el Mètode de Simpson per a aconseguir el resultat desitjat¹.*

¹Aquest càlcul cal lliurar-lo al professor juntament amb la pràctica.

Exercici 3.4.2 *Escriuiu un programa anomenat `normal.c` que preguntí per pantalla h i ϵ i doni el següent “output”:*

- *Un fitxer de sortida anomenat `normal.dat` que contingui la taula de F_X . Aquest fitxer ha de constar de dues columnes. Cada fila contindrà els valors de ih i de $F_X(ih)$ amb $i = 0, 1, 2, \dots, [4/h]$ on $[\cdot]$ denota la part entera d'un nombre. Observem que per $i > 0$,*

$$F_X(ih) = F_X((i-1)h) + \int_x^{x+h} f_X(s) ds.$$

Per tant, els valors de $F_X(ih)$ amb $i > 1$ es calcularan recursivament amb l'ajut del procediment `simpson_normal`. Els valors de $F_X(ih)$ s'escriuran solament amb els decimals significatius en funció de ϵ . Es a dir el nombre de decimals serà exactament el més petit natural més gran o igual que $\log_{10}(2\epsilon)$.

- *Una gràfica de la funció $F_X(x)$ amb $x \in (-4, 4)$. La gràfica es crearà a partir dels punts $F_X(ih)$ units per rectes. La part de la gràfica corresponent a les x negatives es dibuixarà al mateix temps que la corresponent a les x positives usant la propietat de simetria de la funció F_X descrita abans. En efecte: $F_X(-ih) = 1 - F_X(ih)$.*

Capítol 4

Sistemes lineals

4.1 Sistemes d'equacions lineals. Mètode LU.

Volem calcular la matriu inversa A^{-1} per una matriu A no singular, a partir de la descomposició LU de la mateixa. En primer lloc, implementarem la descomposició LU i a continuació farem el càlcul de la inversa.

Exercici 4.1.1 *Implementeu una acció que faci la descomposició LU utilitzant pivotatge parcial per columnes. Per controlar les permutacions es pot utilitzar un vector de permutacions `perm` que contindrà la informació de les possibles permutacions realitzades. La passa de paràmetres ha de ser la següent:*

```
int lu (double **a , int n , int *perm , double tol )
```

Les especificacions són les següents:

- `a` → és un apuntador a les files de la matriu a ; en retornar conté la matriu descomposta. El rang de A va de $a(0,0)$ fins $a(n-1,n-1)$.
- `n` → dimensió d' A .
- `tol` → tolerància usada sobre els pivots per decidir si A es singular o no.
- `perm` → apuntador al primer element d'un vector que indica la fila on hem trobat el pivot en cada pas; `perm[i]=j` vol dir que en el pas i -è el pivot era a la fila j

- `lu` → retorna :
 - 1 si la descomposició ha estat possible en un nombre parell de permutacions.
 - 1 idem però amb un nombre senar.
 - 0 la descomposició no ha estat possible (algun pivot menor que `tol`).

Exercici 4.1.2 En segon lloc, implementeu la resolució d'un sistema $Ax = b$ a partir de la descomposició de la matriu que surt de l'acció anterior `lu`. En aquest pas aplicarem les permutacions que guardem en el vector `perm` al vector de termes independents del sistema. La passa de paràmetres ha de ser la següent:

```
void resol (double **a , double *x , double *b , int n , int *perm )
```

Les especificacions són les mateixes per a les variables que apareixien en l'apartat anterior, i com a noves variables tenim:

- `x` → apuntador al primer element del vector solució `x`.
- `b` → apuntador al primer element del terme independent del sistema.

Exercici 4.1.3 Feu un programa principal que denotarem per `inversa.c` que llegirà una matriu de doubles d'un fitxer amb el mateix format que l'arxiu `inici.c` de la Pràctica 0. El programa ha de demanar per teclat el nom d'un fitxer de dades i ha de d'escriure en un arxiu que anomenarem `inversa.out` la matriu inversa (escrita per files) i el temps de càlcul, que també s'ha de donar per pantalla.

Per comprovar els resultats podeu utilitzar el fitxer de dades `ax_b.dat` que també conté la solució del sistema. La comprovació de la inversa es pot fer directament a partir del producte AA^{-1} .

4.2 Sistemes sobredeterminats. Mètode QR.

Volem calcular la solució per mínims quadrats del sistema d'equacions $Ax = b$, on A és la matriu $m \times n$ d'un sistema sobredeterminat, i b és el vector de termes independents.

Trobarem la solució a partir de la descomposició QR de la matriu del sistema.

Exercici 4.2.1 Es proporciona l'acció `qrhouse.c`, que fa la descomposició QR d'una matriu A que té `num_fil` com a nombre de files i `num_col` com a nombre de columnes.

La seva passada de paràmetres és

```
void qrhouse (double **A, int num_fil, int num_col, double tol);
```

La variable `tol` s'utilitza per decidir si la norma del vector columna corresponent de la matriu A es pot considerar zero.

En sortir, la matriu A haurà estat modificada i contindrà la descomposició QR, amb R a la part tridiagonal superior i els vectors de Householder corresponents a la part tridiagonal inferior.

1. Llegiu les dades del fitxer `sisitema.dat` que tindrà la següent estructura:

A la primera línia, hi haurà un valor enter m que correspon al nombre de files de la matriu, i a la segona línia, un valor enter n que correspon al nombre de columnes de la matriu.

A les $m \times n$ línies següents, els valors de les components de la matriu del sistema A , recorreguda per files. Seguidament, les m components del vector de termes independents b , sense cap línia en blanc.

2. Una vegada llegides les dades, crideu `qrhouse` per obtenir la descomposició QR i a continuació modifiqueu el vector de termes independents de forma adequada.
3. Finalment doneu el temps d'execució, el vector solució x i el residu del problema en norma euclidiana, és a dir,

$$r = \| Ax - b \|_2 .$$

(Observació: Es pot utilitzar la matriu i termes independents originals, o bé els modificats. El resultat hauria de ser el mateix).

4. Repetiu l'exercici amb les dades del fitxer `m25_14.dat`.