ICC Pràctica 1: Àlgebra lineal numèrica

Objectiu: L'objectiu d'aquesta pràctica és implementar en C les rutines bàsiques per resoldre sistemes lineals usant el mètode de Gauss (sense o amb pivotatge) i algunes aplicacions. Donat $n \geq 2$, considerem el sistema lineal $Ax = b$ on $x, b \in \mathbb{R}^n$ i $A \in \mathbb{R}^n \times \mathbb{R}^n$ és una matriu de rang n.

Estructurarem l'enunciat de la pràctica en diferents parts per tal de facilitar-ne la comprensió i la implementació de les funcions. Les diferents parts s'aniran explicant en les successives classes de laboratori d'ordinadors i l'enunciat s'anirà actualitzant progressivament.

Important: No es corregiran les entregues que no segueixin les indicacions indicades al llarg de l'enunciat i/o en les classes de laboratori d'ordinadors.

Organització dels fitxers:

- En començar la pràctica creeu un directori `Cognom1Cognom2Nom_prac1`. Aquest directori contindrà exclusivament
  - Els arxius .c corresponents a la pràctica (amb el nom que s'indica a l'enunciat).
  - L'arxiu `funs_linalg.h` amb les capçaleres de les funcions del fitxer `funs_linalg.c`.
- Les funcions principals estaran en fitxers `main_XXX.c`. La resta de funcions necessàries per compilar i executar els codis relatius a la resolució de sistemes lineals en el fitxer `funs_linalg.c`.

Instruccions per entregar: A la corresponent tasca del Campus Virtual s'entregarà un únic fitxer `Cognom1Cognom2Nom_prac1.tgz` que contindrà *exclusivament*[1]:

- Els arxius .c que s'indiquen al llarg de l'enunciat, i que contindran (exclusivament) el codi C de les funcions programades que s'indiquen per la realització de la pràctica.
- El fitxer `funs_linalg.h` amb les capçaleres de les funcions.
- Un fitxer `.pdf` amb una petita explicació del que s'ha fet, els resultats obtinguts, comprovacions fetes, detalls d'implementacions, respostes dels enunciats, etc.

Per crear l'arxiu `.tgz` executeu des del directori pare la comanda

```
tar -czvf Cognom1Cognom2Nom_prac1.tgz ./Cognom1Cognom2Nom
```

Data (límit) d'entrega: Dia 3 de Novembre de 2021 a les 23:55h.

[1] Assegureu-vos que no hi ha cap altre arxiu en el directori: elimineu objectes, executables, ocults, ...

1 Resolució de sistemes triangulars

1. Implementeu una funció per resoldre el sistema lineal *triangular superior* $Ax = b$.

   - La funció tindrà com a capçalera

     `int resoltrisup(int n, double **A, double *b, double *x, double tol)`

     i rebrà com a paràmetres: la dimensió `n`, la matriu A, el vector b, i la tolerància acceptada `tol`.

   - Suposarà que la matriu A és triangular superior, és a dir, tal que $a_{ij} = 0$ per $0 \leq j < i$, $i = 0, \ldots, n-1$ (no caldrà comprovar-ho, només usarà els valors de la part superior).

   - Resoldrà el sistema usant *substitució enrere*

$$x_{n-1} = b_{n-1}/a_{n-1,n-1}, \qquad x_i = \left(b_i - \sum_{k=i+1}^{n-1} a_{ik}\, x_k \right) / a_{ii} \quad i = n-2, \ldots, 1, 0.$$

   - La solució del sistema es guardarà en el vector `x`. D'altra banda, la funció retornarà 0 si ha pogut resoldre el sistema (si cap $|a_{ii}| < \epsilon$, ϵ tolerància donada) i 1 altrament.

2. Per comprovar la rutina anterior, implementeu una funció principal que cridi la funció `resoltrisup` i escrigui la solució x del sistema i el residu $\|Ax - b\|_2$. La funció principal cridarà les funcions `prodMatVec` i `prod_esc` fetes en les sessions de *Introducció al C* per calcular el residu. Guardeu la funció principal en un fitxer amb nom `main_trisup.c`.

   Useu-la per resoldre els sistemes triangulars següents

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

$$A = \begin{pmatrix} 1.0234 & 2.0981 & 9.9871 & 1.1 \\ 0 & -6.9876 & 2.2222 & 0.3333 \\ 0 & 0 & -1.9870 & 20.121 \\ 0 & 0 & 0 & 1.1234 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

   usant toleràncies 10^{-3}, 10^{-6}, 10^{-9} i 10^{-12}.

3. (Opcional) Feu la funció `resoltriinf` anàloga per a resoldre sistemes triangulars inferiors i implementeu una funció principal per a resoldre $A^\top x = b$, amb A i b de l'exercici anterior.

2 El mètode de Gauss sense pivotatge

4. Implementeu una funció amb capçalera

```
int gauss(int n, double **A, double *b, double tol)
```

que resolgui el sistema $Ax = b$, on $A \in \mathbb{R}^{n\times n}$ és ara una matriu qualsevol, usant el **mètode de Gauss sense pivotatge** (veure teoria per les fórmules corresponents).

- Un cop triangularitzat el sistema cridarà `resoltrisup` per a resoldre-ho.
- A l'entrada de la funció, `b` conté el terme independent b.
- A la sortida de la funció, la part triangular superior de `A` contindrà la matriu triangularitzada resultant d'aplicar Gauss, i `b` contindrá la solució del sistema.
- La funció `gauss` retornarà 0 si ha trobat la solució i 1 altrament.

5. Escriviu una funció principal (`main_gauss.c`) que llegeixi una matriu $A = (a_{ij})$, llegeixi un vector $b = (b_i)$ i cridi `gauss`. Una vegada s'hagi calculat la solució x del sistema $Ax = b$, el programa principal escriurà el vector solució x i el valor del residu $\|Ax-b\|_2$.

   Resoleu, amb diverses toleràncies (10^{-3}, 10^{-6}, 10^{-9} i 10^{-12}), els sistemes següents,

$$A_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ -2 & 1 & 2 & 3 \\ -3 & -2 & 1 & 2 \\ -4 & -3 & -2 & 1 \end{pmatrix}, b_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \qquad A_2 = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{pmatrix}, b_2 = \begin{pmatrix} \frac{25}{12} \\ \frac{77}{60} \\ \frac{19}{20} \\ \frac{319}{420} \end{pmatrix}$$

6. Denotem per A la matriu inicial i per $\tilde{A}$ la matriu que tenim a la sortida de la funció `gauss`. Modifiqueu la funció `gauss` per tal de que $\tilde{A}$ tingui els multiplicadors en la part inferior estricte (i.e. excloent la diagonal) i escribiu una funció

```
double checkLU(int n, double **a, double **acp)
```

que rebi la matriu $\tilde{A}$ que s'obté després d'aplicar `gauss` i la matriu A original.

   Denotem per $L = \mathrm{tril}(\tilde{A}) + \mathrm{Id}$ i per $U = \mathrm{triu}(\tilde{A})$, on $\mathrm{tril}(\tilde{A})$ i $\mathrm{triu}(\tilde{A})$ denoten, respectivament, la part triangular inferior estricte (i.e. excloent la diagonal) i la part triangular superior de $\tilde{A}$. Si $B = A - LU$, la funció retornarà $\max_{0\le i,j\le n} |B_{i,j}|$.

   Comproveu la funció usant els sistemes de l'apartat 4 (el terme independent b no juga cap paper). Guardeu la funció main corresponent en el fitxer `main_gaussLU.c`.

   Per pujar nota: La funció es pot implementar sense crear les matrius L i U, ni la matriu $B = A - LU$. De fet, no cal usar cap matriu ni vector auxiliar.

7. (Opcional) Implementeu una funció principal que resolgui $Ax = e_i$, on e_i denota el vector i-èssim de la base canònica de $\mathbb{R}^n$, usant la descomposició LU de A_2.

3 Mètode de Gauss – pivotatge maximal per columnes

8. Implementeu una funció amb capçalera

```
int gausspiv(int n,double **A, double *v, double tol)
```

que resolgui el sistema $Ax = b$ usant el **mètode de Gauss amb pivotatge**.

- Recordem que, en cada pas k, abans de calcular m_{ik}, es busca $\max_{j=k,\dots,n} |a_{jk}^{(k)}|$. Si denotem per $l \in \{k, k+1, \dots, n\}$ la fila tal que $a_{lk}^{(k)}$ assoleix aquest màxim, llavors s'intercanvia la fila l per la fila k (també l'element corresponent del vector b) i es continua amb el mètode.
- Aquesta funció es una modificació de la funció `gauss`.
- L'intercanvi de files de la matriu en el pivotatge es farà intercanviant els apuntadors corresponents.

9. Modifiqueu la funció principal de l'exercici 4. per tal que cridi la funció `gausspiv` en lloc de la funció `gauss`. Guardeu la funció principal en l'arxiu `main_gausspiv.c`

   Per provar la funció podeu resoldre els sistemes de l'exercici 4. usant les funcions que acabeu de programar i compareu els valors del residu en ambdós casos.

10. Considereu el sistema $Ax = b$ on $A = (a_{i,j})_{1\le i,j\le n}$ és la matriu Hessenberg superior de Frank, que té coeficients

$$a_{i,j} = \begin{cases} 0 & \text{si } j \le i-2 \\ n+1-i & \text{si } j = i-1 \\ n+1-j & \text{si } j \ge i \end{cases}$$

   i $b = Ax_0$ on $x_0 = (1, \dots, 1)^\top$. Feu una funció principal que resolgui el sistema anterior per $n = 2, \dots, 24$, i escrigui en un fitxer el valor de n i el error relatiu de la solució x obtinguda

$$e_{\text{rel}}(x) = \|x - x_0\|_2 / \|x_0\|_2.$$

   Deixarà escollir si s'usa `gauss` o `gausspiv` i escriurà en el fitxer `frankG.out` o `frankGP.out`, respectivament. Guardeu la funció principal en el fitxer `main_frank.c`

   Representeu les gràfiques de $e_{\text{rel}}(x)$ en funció de n usant `gnuplot`. Què podeu dir de la precisió de la solució del sistema en funció de n?

11. (Opcional) Comproveu el temps de resolució de sistemes lineals usant el mètode de Gauss. Per fer-ho, per a cada $2 \le n \le 200$ genereu 10^3 sistemes aleatoris i mesureu el que triga el programa en resoldre el total de sistemes. Representeu la gràfica del temps de còmput en funció de n i justifiqueu el seu comportament.

Mesurar temps de còmput.

Useu la funció `clock()` de `time.h` de la següent manera:

```
#include <time.h>
...
/*Tipus de variable per guardar el nombre de tics del processador*/
clock_t temps;
...
temps=clock();
...
printf("%d␣%e\n",n,(temps/(double)CLOCKS_PER_SEC));
...
```

Cal cridar a `clock_t()` abans i després de resoldre el sistema i incrementar el seu valor per tenir el temps de còmput total.

Generar nombres aleatoris.

Per generar 10^3 sistemes aleatoris useu la funció `rand()` i inicialitzeu el generador de nombres aleatoris en funció del temps del rellotge intern de l'ordinador abans de generar els nombres del sistema. Useu, per exemple, valors aleatoris entre $[0, 1]$, generats de la següent manera:

```
#include <time.h>
...
srand(time(NULL));
for(i=0; i<n; i++) {
         for(j=0;j<n;j++) a[i][j]= rand()/(double)RAND_MAX;
         b[i]= rand()/(double)RAND_MAX;
         }
...
```