ICC Pràctica 1: Àlgebra lineal numèrica

Objectiu: L'objectiu d'aquesta pràctica és implementar en C algunes rutines bàsiques que facin eliminació gaussiana (sense o amb pivotatge) d'una matriu $A \in \mathbb{R}^{n\times n}$. Usarem aquestes rutines per a calcular la factorització LU d'una matriu A, i per a resoldre sistemes lineals $Ax = b$.

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. [1]

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 `linalg.h` amb les capçaleres de les funcions del fitxer `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 l'eliminació gaussiana, resolució de sistemes lineals, etc, estaràn contingudes en el fitxer `linalg.c`.

Instruccions per a entregar:

A la corresponent tasca del Campus Virtual s'entregarà un únic fitxer `Cognom1Cognom2Nom_prac1.tgz` que contindrà *exclusivament*[2]:

- 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 a la realització de la pràctica.
- El fitxer `linalg.h` amb les capçaleres de les funcions.
- Un fitxer `.pdf` amb una breu memòria que exposi el problema, que s'ha implementat per resoldre-ho, que s'ha fet i que no, els resultats obtinguts i les comprovacions fetes, respostes dels enunciats (si s'escau), millores/alternatives de implementació, exercicis extres, etc.[3]

Data (límit) d'entrega: Dv 27 d'octubre de 2023.
No s'acceptaran entregues més enllà del dia **29 d'octubre de 2023 a les 23:55h**.

---

[1] 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.

[2] Assegureu-vos que no hi ha cap altre arxiu en el directori: elimineu objectes, executables, ocults, ...
Per crear l'arxiu `.tgz` executeu des del directori pare la comanda

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

[3] Una entrega dels codis sense la memòria corresponent es considerarà no entregada.

1 Resolució de sistemes triangulars

Implementeu una funció per a resoldre un sistema lineal *triangular inferior* $Lx = b$ i una funció per a resoldre un sistema lineal *triangular superior* $Ux = b$.

Les funcions tindran com a capçalera

```
int resolLinf(int n,double **a, double *b, double *x, double tol,  int ind_diag)
```

```
int resolUsup(int n,double **a, double *b, double *x,double tol)
```

Rebran com a paràmetres: la dimensió `n` de la matriu, la matriu A en `a`, el vector `b` (terme independent) i la tolerància acceptada `tol`. Les funcions suposaran que les matrius tenen la forma correcta i no ho comprovaran, només usaran els valors de la part corresponent (inferior o superior, respectivament). La funció `resolLinf` rep també el paràmetre `ind_diag`. Si `ind_diag` és 1, només accedirà a la part inferior estricta de la matriu `A` i suposarà que $A_{i,i} = 1$ per tot $i = 1, \ldots, n$. La solució del sistema es guardarà en el vector `x`. Les funcions retornaran 0 si ha pogut resoldre el sistema i 1 altrament (p.ex. si $|a_{ii}| <$ `tol` per algun $1 \leq i \leq n$).

Per comprovar les funcions anteriors, implementeu una funció principal, que es desarà en un fitxer amb nom `main_triang.c`. La funció ha de llegir la dimensió n del sistema, la tolerància, un paràmetre per triar si la matriu és la trasposada, la variable `ind_diag`, una matriu triangular inferior i el terme independent. Cridarà la funció `resolLinf` o `resolUsup`, i escriurà la solució x del sistema, el vector residu i la norma-2 del vector residu. Per calcular el residu implementeu una funció auxiliar (que incloureu en `linalg.c`)

```
void residu(int n,double **a, double *b, double *x, double *r)
```

que rebi la dimensió n, la matriu A, el terme independent b i l'aproximació de la solució x, i calculi el vector residu r cridant a la funció `prodMatVec`. Crideu aquesta funció des del programa principal, i calculeu la norma 2 del residu cridant la funció `prod_esc`. Veure les sessions de *Introducció al C* per les funcions `prodMatVec` i `prod_esc`.

Com a aplicació resoleu els sistemes $Lx = b$ i $L^t x = b$ següents, amb tolerància 10^{-12} i amb `ind_diag` $\in \{0, 1\}$.

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 3 & 2 & -1 & 0 \\ 4 & 3 & 2 & 1 \end{pmatrix}, b_1 = \begin{pmatrix} -4 \\ -5 \\ -6 \\ -2 \end{pmatrix}, \ L_2 = \begin{pmatrix} 1.0234 & 0 & 0 & 0 \\ 2.0981 & -6.9876 & 0 & 0 \\ 9.9871 & 2.2222 & -1.9870 & 0 \\ 1.1 & 0.3333 & 20.121 & 1.1234 \end{pmatrix}, b_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

2 Factorització LU sense pivotatge

Considerem una matriu $A \in \mathbb{R}^{n\times n}$ a la qual li aplicarem el procés d'el·liminació gausiana. Implementeu una funció amb capçalera

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

que faci eliminació gaussiana de la matriu A. Rebrà la dimensió `n` d'una matriu A, la matriu A en `a` i la tolerància `tol` i farà $n-1$ passos d'eliminació gaussiana (si es poden fer). A la sortida: la part triangular superior de `a` contindrà la matriu final després de l'eliminació gaussiana, a la part estrictament inferior de `a` s'hauran guardat els multiplicadors corresponents.

Escriviu una rutina principal (`main_LUsolver.c`) que llegeixi la dimensió n, la tolerància, una matriu $A \in \mathbb{R}^{n\times n}$, un vector $b \in \mathbb{R}^n$ i resolgui el sistema $Ax = b$ usant la descomposició LU de A sense pivotatge. Cridarà les funcions `lu`, `resolLinf`, `resolUsup` i `residu` per a calcular la solució x del sistema $Ax = b$ i el valor del residu $\|Ax - b\|_2$. Escriurà la solució, el vector residu i la seva norma-2.

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

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

3 Factorització PA=LU

Implementeu una funció amb capçalera

```
int palu(int n,double **a,int *p,double tol)
```

que, donada una matriu $A \in \mathbb{R}^{n\times n}$, faci eliminació gaussiana amb pivotatge parcial (o pivotatge maximal per columnes) de la matriu. Aquesta funció és una petita modificació de la funció `lu`:

i) El vector $p \in \mathbb{R}^n$ retorna la permutació P de files: la fila i de PA és la fila $p[i]$ de A. A l'inici $p_i = i$, $1 \le i \le n$.

ii) En cada pas k, abans de calcular el multiplicador m_{ik}, es busca $l \in \{k, k+1, \ldots, n\}$ tal que $a_{lk}^{(k)} = \max\limits_{j=k,\ldots,n} |a_{jk}^{(k)}|$. Llavors s'intercanvien la fila l i la fila k [4] , s'actualitza el vector de permutació intercanviant p_l i p_k, i es continua amb el procés d'eliminació gaussiana.

iii) La funció retorna 0 si no s'ha pogut fer la factorització i retorna la paritat (o signe) ± 1 de la permutació P. La paritat de la permutació és igual a $(-1)^{nc}$ on nc és el nombre de intercanvis de files realitzats en el procès d'eliminació gaussiana amb pivotatge parcial.

Per a comprovar aquesta funció feu una rutina principal `main_PALU.c` que llegeixi la dimensió, la tolerància, una matriu, invoqui a la funció `palu` i escrigui el vector de permutació, la paritat de la permutació i el producte LU.[5]

Aplicacions:

a) Implementeu una funció principal `main_PALUsolver.c` que resolgui sistemes lineals $Ax = b$ usant la factorització $PA = LU$. Cridarà la funció `resolLinf` per a resoldre $L\,y = P\,b$, i després cridarà la funció `resolUsup` per a resoldre el sistema triangular superior $U\,x = y$ i trobar $x \in \mathbb{R}^n$. El programa escriurà el vector p. la norma infinit de $PA - LU$, la norma-2 del vector residu i la solució x. Useu-lo per resoldre els sistemes lineals de l'apartat b) de la secció anterior.

b) Feu un programa principal `main_temps_PALU.c` que per cada n, $1 \le n \le 100$, generi de $N = 1000$ sistemes $Ax = b$ aleatoris amb $a_{ij}, b_i \in [-1, 1]$, i els resolgui usant la factorització $PA = LU$. El programa escriurà un fitxer amb 3 columnes. La 1a columna serà la dimensió n. La 2a columna serà el màxim de les normes-2 dels residus obtinguts en resoldre els N sistemes de dimensió n. La 3a columna serà el temps de còmput promig de resoldre els N sistemes de dimensió n. Representeu les gràfiques corresponents a la norma del residu en funció de n i del temps de còmput en funció de n. Expliqueu els resultats obtinguts.

[4] S'intercanvien tots els elements de les files (inclosos els multiplicadors), es poden intercanviar els punters a fila.

[5] El producte LU es pot fer sense usar cap matriu adicional.

A Exemple de codi per generar valors reals aleatoris

```
/* Es generen valors reals aleatoris en un interval de reals */

#include <stdio.h>      /* printf, scanf */
#include <stdlib.h>     /* rand, srand, RAND_MAX */
#include <time.h>       /* time */

int main(void) {
   int n, k;
   double a, b, x;

   printf("quantitat de valors ?  \n");
   scanf("%d", &n);

   printf("extrems de l'interval de reals ?\n");
   scanf("%le %le", &a, &b);

   printf("reals a [%+20.15e, %+20.15e] \n", a, b);

   srand(time(NULL));          /* genera la llavor inicial */

   for (k=1; k<=n; k++) {
      x = a+(b-a)*((1.*rand())/RAND_MAX);
      printf(" %+20.15e \n", x);
   }

   return 0;
}
```

B Llistat de funcions a implementar

Funcions d'àlgebra lineal a `linalg.c`:

- `resolLinf`
- `resolUsup`
- `residu`
- `lu`
- `palu`
- `prodMatVec`
- `prod_esc`

Mains:

- `main_triang.c`
- `main_LUsolver.c`
- `main_PALUsolver.c`
- `main_temps_PALU.c`
- `main_PALU.c`