OPERACIONS AMB MATRIUS I VECTORS EN C

OBJECTIUS

- Tenir una primera aproximació al llenguatge de programació C.
- Desenvolupar un programa que permeti realitzar operacions algebraiques amb matrius i vectors.
- Compilar i executar un programa utilitzant la línia de comandes i el compilador gcc.

MATERIAL

- Sistema Operatiu Linux
- Compilador gcc

Important!

- La pràctica es farà en **grups** de **dues persones**.
- La **pràctica** consta de 3 **sessions** i el **termini màxim de lliurament** serà **4 dies després de la tercera sessió**.
- A part dels **fitxers de codi**, serà necessari lliurar un **informe de la pràctica**, en el que s'expliqui la feina feta durant les 3 sessions.
- Els arxius de codi i l'informe poden lliurar-se en un arxiu comprimit ZIP, ***identificat amb el NIU i el grup dels estudiants*** o en dos arxius separats (codi i informe).
- En cas de detectar **còpies,** el grup tindrà automàticament un **0 en la nota de pràctiques**

1. INTRODUCCIÓ

En el camp de l'enginyeria de dades moltes vegades ens trobarem amb datasets de grans dimensions, de l'ordre de Gigues o Teres, els quals contenen milers de registres que poden incloure centenars de característiques o camps diferents.

Amb l'objectiu de fer un processament eficient de les dades, una alternativa és reduir les dimensions del dataset, eliminant els registres o camps que no siguin necessaris durant l'etapa de processament.

Una forma simple i molt utilitzada de reduir les dimensions el dataset és aplicar tècniques de factorització de matrius. La factorització de matrius té aplicacions enormes en el camp de l'enginyeria de dades i les seves branques, com podria ser en la intel·ligència artificial o l'aprenentatge automàtic.

Atesa la importància que té l'àlgebra lineal a l'enginyeria de dades, durant el transcurs d'aquesta pràctica es desenvoluparà una llibreria d'àlgebra lineal, que contindrà les operacions matricials utilitzades en algoritmes de machine learning.

Amb l'objectiu de desenvolupar una llibreria eficient que permeti el còmput de matrius de grans dimensions, s'utilitzarà el llenguatge de programació C, que permet gestionar la memòria al desenvolupador i utilitzar la gestió dinàmica de memòria.

2. CONCEPTES PREVIS

Abans de fer la pràctica és important que repassi les operacions bàsiques d'àlgebra lineal amb matrius per poder-les implementar correctament.

Tot i que per a la realització de la pràctica no es requereix un coneixement avançat de programació en el llenguatge de programació C, si no s'ha programat mai en aquest llenguatge, és recomanable consultar prèviament documentació sobre això per poder aprofitar la sessió de pràctiques.

3. EXERCICIS PRÀCTICS

L'objectiu de la pràctica serà realitzar una biblioteca que permeti realitzar operacions matricials/vectorials utilitzant el llenguatge de programació C. La llibreria haurà de permetre realitzar les operacions següents:

- **Multiplicació Escalar**: multiplicar un vector per un escalar.
- **Producte escalar**: producte punt entre dos vectors.
- **Magnitud**: calcular la magnitud d'un vector.
- **Ortogonals**: determinar si dos vectors són ortogonals.
- **Projecció**: calcular la projecció d'un vector sobre un altre.
- **Norma de Frobenius**: calcular la norma de Frobenius d'una matriu.
- **Infini-norma**: calcular la infini-norma d'una matriu.
- **Norma-u:** calcular la norma ú d'una matriu**.**
- **Diagonal dominant:** determinar si una matriu és o no diagonal dominant.
- **Resoldre sistemes d'equacions lineals:** amb un mètode iteratiu.

Com estem fent el desenvolupament d'aquesta biblioteca, cal tenir un conjunt de proves per validar la funcionalitat de cada funció. Per fer això, a part de les funcions anteriors, crearem una matriu i un conjunt de vectors i una funció d'inicialització per aquestes estructures de dades.

Les estructures i constants que declarareu són les següents:

- Constant N igual a 512.
- Matrius (Mat i MatDD) de números reals (punt flotant de precisió simple) de NxN elements.
- Vectors (V1, V2, V3 i V4) de números reals de N elements.

Amb l'objectiu que tots els alumnes utilitzin les mateixes dades d'entrada, a l'hora de generar els elements que contindran la matriu i els vectors, la funció initData haurà d'utilitzar les funcions incloses a l'estàndard de C rand() i srand() per generar valors dels elements. La funció rand() permet generar números aleatoris utilitzant un generador de nombres aleatoris. La funció srand() permet instanciar la llavor que utilitzarà com a base el generador de nombre aleatoris. Per això, passarem com a entrada la llavor "4422543" (srand(4422543)). Amb això aconseguirem generar números pseudo-aleatoris, que seguiran la mateixa seqüència en cada execució del programa. Aquesta funció requereix ser cridada abans de la funció rand().

La funció d'inicialització la proporcionem ja programada (tots heu de fer servir aquesta funció per poder comprovar que els resultats són correctes):

```
void InitData(){
 int i,j;
 srand(8824553);
 for( i = 0; i < N; i++ )
  for( j = 0; j < N; j++ ){
   Mat[i][j]=(((i*j)%3)?-1:1)*(100.0*(rand()/(1.0*RAND_MAX)));
   if ( (abs(i - j) <= 3) && (i != j))
    MatDD[i][j] = (((i*j)%3) ? -1 : 1)*(rand()/(1.0*RAND_MAX));
   else if ( i == j )
    MatDD[i][j]=(((i*j)%3)?-1:1)*(10000.0*(rand()/(1.0*RAND_MAX)));
   else MatDD[i][j] = 0.0;
  }

 for( i = 0; i < N; i++ ){
  V1[i]=(i<N/2)?(((i*j)%3)?-1:1)*(100.0*(rand()/(1.0*RAND_MAX))):0.0;
  V2[i]=(i>=N/2)?(((i*j)%3)?-1:1)*(100.0*(rand()/(1.0*RAND_MAX))):0.0;
  V3[i]=(((i*j)%5)?-1:1)*(100.0*(rand()/(1.0*RAND_MAX)));
 }
}
```

Un cop s'han inicialitzat les estructures, desenvoluparem les funcions següents:

1. Amb l'objectiu de comprovar que el nostre codi funciona, desenvoluparem una funció que mostri per pantalla un cert nombre d'elements (*numel*) d'un vector a partir d'una posició donada (*from*).
   La capçalera d'aquesta funció serà:

   void PrintVect(float vect[N], int from, int numel)

2. Amb el mateix objectiu, desenvoluparem una funció que mostri per pantalla un cert nombre d'elements (*numel*) d'una fila (row) d'una matriu a partir d'una posició donada (*from*).
   La capçalera d'aquesta funció serà:

   void PrintRow(float mat[N][N], int row, int from, int numel)

3. Calcular la multiplicació d'un escalar per un vector (Multiplicació Escalar). Aquesta operació consisteix en multiplicar tots els elements d'un vector per una constant, deixant el resultat en un nou vector.

$$\overrightarrow{vr} = \alpha \times \vec{v}$$

   La capçalera d'aquesta funció serà:

   void MultEscalar(float vect[N], float vectres[N], float alfa)

4. Calcular el producte escalar (o punt) de dos vectors. Aquesta operació calcula la magnitud de la projecció d'un vector *u* sobre un altre vector *v* (multiplicada per la magnitud de *v*) amb la següent operació:

$$\vec{u}.\vec{v} = \sum_{i=0}^{n} u_i \times v_i$$

   La capçalera d'aquesta funció serà:

   float Scalar(float vect1[N], float vect2[N])

5. Calcular la magnitud d'un vector. Aquesta operació calcula la longitud d'un vector fent la següent operació:

$$|\vec{v}| = \sqrt{\sum_{i=0}^{n} v_i^2} = \sqrt{\vec{v}.\vec{v}}$$

   La capçalera d'aquesta funció serà:

float Magnitude(float vect[N])

6. Determinar si dos vectors són ortogonals. Dos vectors són ortogonals si l'angle entre ells és recte, en aquest cas la projecció d'un vector sobre l'altre tindrà magnitud 0. Així, dos vectors són ortogonals si:

$$\vec{u}.\vec{v} = 0$$

La capçalera d'aquesta funció serà:

int Ortogonal(float vect1[N], float vect2[N])

Nota: la funció retorna 1si els vectors són Ortogonals i 0 si no.

7. Calcular el vector projecció d'un vector sobre un altre. La projecció ortogonal d'un vector *u* sobre un vector *v* es calcula amb la següent expressió:

$$P_{\vec{v}}(\vec{u}) = \frac{\vec{u}.\vec{v}}{|\vec{v}|} \times \vec{v}$$

La capçalera d'aquesta funció serà:

void Projection(float vect1[N], float vect2[N], float vectres[N])

8. Calcular la Infini-norma d'una matriu. La infini-norma d'una matriu A ∊ $R_{n\times n}$ es defineix com el màxim de les sumes dels valors absoluts dels elements de cada fila:

$$\|A\|_\infty = \max_{i=0,..,n-1}\left\{\sum_{j=0}^{n-1}\left|a_{i,j}\right|\right\}$$

La capçalera d'aquesta funció serà:

float Infininorm(float M[N][N])

Nota: per calcular el valor absolut d'un real utilitzem la funció *fabs*.

9. Calcular la norma-ú d'una matriu. La norma-ú d'una matriu A ∊ $R_{n\times n}$ es defineix com el màxim de les sumes dels valors absoluts dels elements de cada columna:

$$\|A\|_1 = \max_{j=0,..,n-1} \left\{ \sum_{i=0}^{n-1} |a_{i,j}| \right\}$$

La capçalera d'aquesta funció serà:

float Onenorm(float M[N][N])

10. Calcular la norma de Frobenius d'una matriu. La norma de Frobenius d'una matriu A ∈ $R_{n \times n}$ es defineix com l'arrel quadrada de la suma dels quadrats del coeficients de la matriu:

$$\boldsymbol{Frob} = \sqrt[2]{\sum_{i=0}^{N} \sum_{j=0}^{N} \boldsymbol{a_{i,j}^2}}$$

La capçalera d'aquesta funció serà:

float NormFrobenius(float M[N][N])

11. Determinar si una matriu és o no Diagonal Dominant. Una matriu és diagonal dominant si el valor absolut de l'element de la diagonal és més gran o igual que la suma dels valors absoluts dels altres elements de la fila, per a totes les files de la matriu:

$$|a_{i,i}| \geq \sum_{j \neq i} |a_{i,j}| \ \forall i \ (0 \leq i \ < n)$$

La capçalera d'aquesta funció serà:

int DiagonalDom(float M[N][N])

Nota: la funció retorna 1si la matriu M és diagonal dominant i 0 si no.

12. Resoldre sistemes d'equacions lineals. Per a solucionar sistemes d'equacions lineals de la forma:

$$\begin{pmatrix} a_{11} & . & . & . & a_{1n} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ a_{m1} & . & . & . & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ . \\ . \\ . \\ c_m \end{pmatrix}$$

(o **Ax = b** per simplificar), podem utilitzar, en certes condicions, mètodes iteratius com el de *Jacobi*. Aquets mètodes proposen descompondre la matriu A en S – T, transformant el sistema **Ax = b** en **Sx = Tx + b**, del qual pot derivar-se $\mathbf{Sx_{k+1} = Tx_k + b}$ sempre que es compleixi:

- **S** ha de ser senzilla (diagonal o triangular) i invertible
- la successió $\mathbf{x_k}$ ha convergir a la solució

En particular, el mètode de *Jacobi* proposa que **S** sigui la diagonal d'**A**, així:

$$
\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{1n}x_n)/a_{22} \\
&\ldots \\
&\ldots \\
x_n &= (b_n - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1n-1}x_{n-1})/a_{nn}
\end{aligned}
$$

Pel que, donada una solució inicial qualsevol, els passos de refinament es calculen com:

$$x_i^{k+1} = \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{n} a_{ij} x_j^k \right] \Big/ a_{ii}$$

El mètode pot aplicar-se un nombre d'iteracions determinat o bé fins que la diferència entre dos solucions consecutives estigui per sota d'un cert llindar.

El mètode de Jacobi convergeix si la matriu A és **estrictament irreductible** (diagonal dominant).

La capçalera d'aquesta funció serà:

int Jacobi(float M[N][N] , float vect[N], float vectres[N], unsigned iter)

Nota: la funció retorna 1si el mètode de Jacobi es pot aplicar i 0 si no.

Un cop programades les funcions, fareu servir els vectors i matrius inicialitzats en la funció InitData per realitzar el següent conjunt de proves de validació en la funció main del programa:

A. La primera funció que invocareu serà InitData. Per veure quina forma tenen les matrius i vectors creats visualitzareu:
   a. **Els elements 0 al 9 i 256 al 265 dels vectors V1, V2 i V3.**
      Resultat:

V1 del 0 al 9 i del 256 al 265:
69.566933 -89.503113 -39.197334 19.066200 -9.117732 -75.454338 96.059006 -25.927046 -80.511101 94.021248
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
V2 del 0 al 9 i del 256 al 265:
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
-5.451517 -24.667860 82.542519 -16.721731 -2.251943 81.090149 -20.380577 -82.087517 41.173542 -74.244781
V3 del 0 al 9 i del 256 al 265:
35.603161 -0.445214 -89.679192 -50.968040 -67.174515 20.219330 -21.083359 -52.133034 -33.070210 -6.267238
-13.743254 -44.734402 -0.750853 -71.902992 75.043221 -28.843531 -13.066498 -93.570541 -7.777743 20.148510

B. Els elements 0 al 9 de les files 0 i 100 de la matriu Mat.

Resultat:

Mat fila 0 i fila 100 del 0 al 9:
16.374699 95.241478 63.198044 94.032303 17.638218 64.790344 51.421196 98.063591 31.335430 75.562134
7.662778 -6.477802 -42.113594 53.627747 -83.585808 -83.037819 93.882782 -25.798609 -4.227375 21.905283

C. Els elements 0 al 9 de la fila 0 i 90 a 99 de la fila 100 de la matriu *MatDD.*

Resultat:

MatDD fila 0 del 0 al 9 i fila 100 del 95 al 104:
6211.223633 0.000490 0.532091 0.093330 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 -0.055969 -0.183933 0.076992 -1257.427368 -0.937005 0.132788 -0.465951 0.000000

D. Per cada matriu (Mat i MatDD) calculareu i visualitzareu:

a. La seva Infini-norma
b. La seva norma ú
c. La seva norma de Frobenius
d. Si la matriu és o no diagonal dominant

Resultat:

Infininorma de Mat = 27798.012
Norma ú de Mat = 27684.238
Norma de Frobenius de Mat = 29581.977
La matriu Mat no és diagonal dominant

Infininorma de MatDD = 9993.846
Norma ú de MatDD = 9992.489
Norma de Frobenius de MatDD = 132122.969
La matriu MatDD és diagonal dominant

E. Calculareu i visualitzareu els següents productes escalars:

a. V1 ·V2
b. V1 ·V3
c. V2·V3

Resultat:

Escalar <V1,V2> = 0.000000 Escalar <V1,V3> = 156787.875000 Escalar <V2,V3> = 160848.625000

F. Calculareu i visualitzareu la magnitud de V1, V2 i V3

Resultat:

Magnitud V1,V2 i V3 = 913.547058 914.269592 1305.615723

G. Calculareu i visualitzareu si V1 és o no ortogonal amb V2 i V3 i si V2 i V3 són ortogonals (totes les combinacions possibles).

Resultat:

V1 i V2 són ortogonals

H. Calculareu la multiplicació del vector V3 amb l'escalar 2.0 i visualitzareu els elements 0 al 9 i 256 al 265 del vector resultant.

Resultat:

Els elements 0 al 9 i 256 al 265 del resultat de multiplicar V3x2.0 són:
71.206322 -0.890428 -179.358383 -101.936081 -134.349030 40.438660 -42.166718 -104.266068 -66.140419 -12.534475
-27.486507 -89.468803 -1.501705 -143.805984 150.086441 -57.687061 -26.132996 -187.141083 -15.555487 40.297020

I. Calculareu la projecció del V2 sobre V3 i visualitzareu els 10 primers elements del vector resultant. Repetiu per la projecció de V1 sobre V2.

Resultat:

Els elements 0 a 9 del resultat de la projecció de V2 sobre V3 són:
4386.221191 -54.849236 -11048.254883 -6279.136230 -8275.734375 2490.971436 -2597.417725 -6422.660645 -4074.167969 -772.108154
Els elements 0 a 9 del resultat de la projecció de V1 sobre V2 són:
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

J. Calculareu la solució del sistema d'equacions MatDD*X = V3. La qual cosa vol dir que estaríeu trobant una aproximació a una solució d'un sistema de 512 equacions amb 512 incògnites. Feu un primer càlcul amb 1 iteració i un segon amb 1000. Visualitzeu els 10 primers elements de la solució en ambdós casos. Què passa si el sistema que volem resoldre és Mat.X = V3?

Resultat:

Els elements 0 a 9 de la solució (1 iter) del sistema d'equacions són:
0.005732 0.000091 0.013348 -0.006099 7.050530 -0.003038 -0.010917 0.005360 0.005016 -0.003713
Els elements 0 a 9 de la solució (1000 iters) del sistema d'equacions són:
0.005732 -0.001053 0.012309 -0.006581 7.048634 -0.003627 -0.013397 0.005094 0.005008 -0.003718
La matriu M no és diagonal dominant, no es pot aplicar Jacobi

Punt Extra:

Heu programat el mètode de Jacobi per l'aproximació a la solució d'un sistema d'equacions lineal (Ax = b), però, com sabeu que tan bona és la solució que hem trobat amb aquest mètode? Dit d'una altra manera, la funció ha calculat un vector x que se suposa que satisfà MatDD*x = V3, però tots sabem que la solució trobada és una aproximació a la solució real. Sense haver de calcular la solució exacte, com podem saber que tan bona és la solució trobada?

- Descriviu matemàticament quin seria el procediment a seguir per respondre la pregunta (+0.25)
- Programeu la vostra solució i indiqueu el resultat pels dos casos provats en el punt J (+0.75)

4. AVALUACIÓ

Per a l'avaluació de la pràctica es tindran en compte els elements següents:

1. **Funcionament**. Si no s'aconsegueixen totes les funcionalitats demanades, es reduirà la nota final. Aquest serà l'apartat de més pes (45%)

2. **Memòria**. Que ha d'incloure la descripció del treball realitzat (anàlisi del problema i disseny de la solució), així com de les dificultats més importants que us hàgiu trobat (20%)
3. **Qualitat de la solució**. Es valorarà la qualitat del codi desenvolupat (ús del llenguatge C, organització de la solució, documentació i presentació) (15%)
4. **Assistència a les sessions de pràctiques i participació en el seguiment realitzat pel professor de pràctiques. (20%)**

Aquest curs hem introduït l'ús del programari de control de versions git i la seva versió al núvol GitHub. Amb l'objectiu d'animar-vos a provar-ne l'ús, hem decidit que aquells grups que demostrin que han utilitzat aquesta plataforma de forma correcta durant el desenvolupament de tots els exercicis pràctics de l'assignatura seran premiats amb un punt extra a la part pràctica de l'assignatura.

Aquest punt no es podrà utilitzar en cap cas per aprovar les pràctiques (si no s'arriba al 5, les pràctiques estaran suspeses independentment que s'hagi utilitzat correctament git i GitHub).

Perquè el vostre professor de pràctiques pugui valorar si s'ha utilitzat correctament git + GitHub haureu d'incloure a la memòria de cada exercici l'enllaç al repositori de la pràctica a GitHub.