Documentació Mètodes Numèrics 1

Ignasi Sánchez Rodríguez

December 14, 2015

Abstract

Aquesta és la documentació per al fitxer *mn.h* que contè funcions útils per a la assignatura *Mètodes Numèrics I* del grau de Matemàtiques de la Universitat de Barcelona.

Contents

1 Assignació de memòria

1.1 double** dmallocm(int n)

Descripció: Assigna memòria per a una matriu de dimensió nxn de tipus double.
Paràmetres:

- *int:* Dimensió de la matriu.

Retorna: Una matriu nxn de tipus double.

1.2 double* dmallocv(int n)

Descripció: Assigna memòria per a un vector de dimensió n de tipus double.
Paràmetres:

- *int:* Dimensió del vector.

Retorna: Un vector de dimensió n de tipus double.

1.3 int** imallocm(int n)

Descripció: Assigna memòria per a una matriu de dimensió nxn de tipus int.
Paràmetres:

- *int:* Dimensió de la matriu.

Retorna: Una matriu nxn de tipus int.

1.4 int* imallocv(double **a, int n)

Descripció: Assigna memòria per a un vector de dimensió n de tipus int.
Paràmetres:

- *int:* Dimensió del vector.

Retorna: Un vector de dimensió n de tipus int.

1.5 void dfreem(int **a, int n)

Descripció: Llibera memòria d'una matriu nxn-dimensional de tipus double.
Paràmetres:

- *double**:* La matriu.
- *int:* Dimensió de la matriu.

Retorna: —

1.6 void ifreem(int **a, int n)

Descripció: Llibera memòria d'una matriu nxn-dimensional de tipus int.
Paràmetres:

- *int**:* La matriu.
- *int:* Dimensió de la matriu.

Retorna: —

2 Inicialitzacio de matrius i vectors

2.1 double** identity_matrix(int n)

Descripció: Assigna i omple una matriu identitat nxn-dimensional (Una matriu identitat és una matriu de la forma $A = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$)

Paràmetres:

- *int:* Dimensió de la matriu.

Retorna: Una matriu identitat de dimensió nxn.

2.2 double** drandom_matrix(int n, double min, double max)

Descripció: Assigna i omple una matriu nxn-dimensional amb valors double aleatoris.
Paràmetres:

- *int:* Dimensió de la matriu.
- *double:* El valor mínim.
- *double:* El valor màxim.

Retorna: Una matriu de dimensió nxn amb valors double aleatoris.

2.3 int** irandom_matrix(int n, int min, int max)

Descripció: Assigna i omple una matriu nxn-dimensional amb valors int aleatoris.
Paràmetres:

- *int:* Dimensió de la matriu.
- *int:* El valor mínim.
- *int:* El valor màxim.

Retorna: Una matriu de dimensió nxn amb valors int aleatoris.

2.4 double* natural_base_vec(int n, unsigned int j)

Descripció: Assigna i omple el j-èssim vector de la base canònica (el vector j-èssim de la base canònica és de la forma $(0, \dots, 0, \overset{(j)}{1}, 0, \dots, 0)$)
Paràmetres:

- *int:* Dimensió del vector.
- *unsigned int:* L'índex del 1.

Retorna: El j-èssim vector n-dimensional de la base canònica.

2.5 double* drandom_vector(int n, double min, double max)

Descripció: Assigna i omple un vector n-dimensional amb valors double aleatoris.
Paràmetres:

- *int:* Dimensió del vector.
- *double:* El valor mínim.
- *double:* El valor màxim.

Retorna: Un vector n-dimensional amb valors double aleatoris.

2.6 int* irandom_vector(int n, int min, int max)

Descripció: Assigna i omple un vector n-dimensional amb valors int aleatoris.
Paràmetres:

- *int:* Dimensió del vector.
- *int:* El valor mínim.
- *int:* El valor màxim.

Retorna: Un vector n-dimensional amb valors int aleatoris.

3 Imprimir i administració de fitxers

3.1 void printe(char* str)

Descripció: Imprimeix el missatge donat i surt del programa amb un codi d'error 1.
Paràmetres:

- *char*:* Un missatge per imprimir.

Retorna: —

3.2 void dprintm(double **a, int n)

Descripció: Imprimeix una matriu nxn-dimensional de valors double.
Paràmetres:

- *double**:* La matriu a imprimir.
- *int:* Dimensió de la matriu.

Retorna: —

3.3 void iprintm(int **a, int n)

Descripció: Imprimeix una matriu nxn-dimensional de valors int.
Paràmetres:

- *int**:* La matriu a imprimir.
- *int:* Dimensió de la matriu.

Retorna: —

3.4 void dprintv(double *v, int n)

Descripció: Imprimeix un vector n-dimensional de valors double.
Paràmetres:

- *double*:* El vector a imprimir.
- *int:* Dimensió del vector.

Retorna: —

3.5 void iprintv(int *v, int n)

Descripció: Imprimeix un vector n-dimensional de valors int.
Paràmetres:

- *int*:* El vector a imprimir.
- *int:* Dimensió del vector.

Retorna: —

3.6 void fdprintm(double **a, FILE *f, int n)

Descripció: Imprimeix una matriu nxn-dimensional de valors double a un fitxer.
Paràmetres:

- *double**:* La matriu a imprimir.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* Dimensió de la matriu.

Retorna: —

3.7 void fiprintm(int **a, FILE *f, int n)

Descripció: Imprimeix una matriu nxn-dimensional de valors int en un fitxer.
Paràmetres:

- *int**:* La matriu a imprimir.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* Dimensió de la matriu.

Retorna: —

3.8 void fdprintv(double *v, FILE *f, int n)

Descripció: Imprimeix un vector n-dimensional de valors double en un fitxer.
Paràmetres:

- *double*:* El vector a imprimir.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* Dimensió del vector.

Retorna: —

3.9 void fiprintv(int *v, FILE *f, int n)

Descripció: Imprimeix un vector n-dimensional de valors int en un fitxer.
Paràmetres:

- *int*:* El vector a imprimir.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* Dimensió del vector.

Retorna: —

3.10 void printPALU(double **lu, int *p, int n)

Descripció: Imprimeix la següent informació sobre la descomposició $PA = LU$:

- El vector de permutacions p.
- La matriu triangular inferior amb uns a la diagonal L.
- La matrius triangular superior U.
- La matriu LU.
- El producte $L * U$

Paràmetres:

- *double**:* La matriu LU.
- *int*:* El vector de permutacions.
- *int:* La dimensió de la matriu i del vector.

Retorna: —

3.11 void printLU(double **lu, int n)

Descripció: Imprimeix la següent informació sobre la descomposició $A = LU$:

- La matriu triangular inferior amb uns a la diagonal L.
- La matrius triangular superior U.
- La matriu LU.
- El producte $L * U$

Paràmetres:

- *double**:* La matriu LU.
- *int:* La dimensió de la matriu i del vector.

Retorna: —

3.12 void fprintPALU(double **lu, int *p, FILE *f, int n)

Descripció: Imprimeix la següent informació sobre la descomposició $PA = LU$ en un fitxer:

- El vector de permutacions p.
- La matriu triangular inferior amb uns a la diagonal L.
- La matrius triangular superior U.
- La matriu LU.
- El producte $L * U$

Paràmetres:

- *double**:* La matriu LU.
- *int*:* El vector de permutacions.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* La dimensió de la matriu i del vector.

Retorna: —

3.13 void fprintLU(double **lu, FILE *f, int n)

Descripció: Imprimeix la següent informació sobre la descomposició $A = LU$ en un fitxer:

- La matriu triangular inferior amb uns a la diagonal L.
- La matrius triangular superior U.
- La matriu LU.
- El producte $L * U$

Paràmetres:

- *double**:* La matriu LU.
- *FILE*:* El fitxer on volem imprimir. Ha d'estar obert.
- *int:* La dimensió de la matriu i del vector.

Retorna: —

3.14 void OPEN_IN_FILE(FILE **f)

Descripció: Obre el fitxer **f com un fitxer d'entrada amb el nom del fitxer en el que estem treballant. Per exeple, si estem treballant en *exemple.c*, el fitxer d'entrada que obrirà serà el *exemple.in*: *Paràmetres:*

- *FILE**:* La referència al fitxer que volem obrir.

Retorna: —

3.15 void OPEN_OUT_FILE(FILE **f)

Descripció: Obre el fitxer **f com un fitxer de sortida amb el nom del fitxer en el que estem treballant. Per exeple, si estem treballant en *exemple.c*, el fitxer de sortida que obrirà serà el *exemple.out*:
Paràmetres:

- *FILE**:* La referència al fitxer que volem obrir.

Retorna: —

3.16 void file_open(FILE** f, const char* name, int inout)

Descripció: Usada en les definicions 3.14 i 3.15. És recomanable no usar-la.
Paràmetres:

- *FILE**:* La referència al fitxer que volem obrir
- *char*:* El nom que volem donar-li al fitxer. El fitxer s'anomenarà *name.in* o *name.out* depenent del paràmetre *inout*.
- *int:* El valor d'aquest paràmetre determina si és un fitxer d'entrada o de sortida. Si $inout == 0$, aleshores és un fitxer d'entrada. Si $inout > 0$ aleshores és un fitxer de sortida.

Retorna: —

4 Operacions amb matrius

4.1 double LU_factorization(double **a, int n)

Descripció: Calcula la factorització *LU* de la matriu nxn-dimensional si és possible.
Paràmetres:

- *double**:* La matriu a factoritzar.
- *int:* Dimensió de la matriu.

Retorna: El determinant de la matriu.

4.2 double PALU_factorization(double **a, int *p, int n)

Descripció: Calcula la factorització *LU* amb permutació de la matriu nxn-dimensional si és possible.
Paràmetres:

- *double**:* La matriu a factoritzar.
- *int*:* El vector de permutacions.
- *int:* Dimensió de la matriu i del vector.

Retorna: El determinant de la matriu.

4.3 double** LU_product(double **lu, int n)

Descripció: El producte de la matriu LU.
Paràmetres:

- *double**:* La matriu a factoritzar.
- *int:* Dimensió de la matriu.

Retorna: La matriu producte.

4.4 double infinity_norm(double **a, int n)

Descripció: Calcula la norma infinit $\| \|_\infty$ de la matriu nxn-dimensional.
Paràmetres:

- *double**:* La matriu de la que volem trobar la norma.
- *int:* Dimensió de la matriu.

Retorna: El determinant de la matriu.

4.5 void solve_LUx(double **lu, double *b, int n)

Descripció: Resol el sistema lineal $LUx = b$.
Paràmetres:

- *double**:* La matriu LU.
- *double*:* El vector de termes independents.
- *int:* Dimensió de la matriu i del vector.

Retorna: —

4.6 void solve_PLUx(double **lu, double *b, int *p, int n)

Descripció: Resol el sistema lineal $LUx = Pb$, on LU és una matriu factoritzada amb permutació.
Paràmetres:

- *double**:* La matriu LU.
- *double*:* El vector de termes independents.
- *int*:* El vector de permutacions.
- *int:* Dimensió de la matriu i dels vectors.

Retorna: —

4.7 void triinf(double **lu, double *b, int n)

Descripció: Resol el sistema lineal $Lx = b$, on L és una matriu triagular inferior amb uns a la diagonal.
Paràmetres:

- *double**:* La matriu L.
- *double*:* El vector de termes independents.
- *int:* Dimensió de la matriu i del vector.

Retorna: —

4.8 void trisup(double **lu, double *b, int n)

Descripció: Resol el sistema lineal $Ux = b$, on U és una matriu triagular superior.
Paràmetres:

- *double**:* La matriu U.
- *double*:* El vector de termes independents.
- *int:* Dimensió de la matriu i del vector.

Retorna: —

4.9 void permute_b(double *b, int *p, int n)

Descripció: Permuta el vector b segons el vector de permutacions p.
Paràmetres:

- *double*:* El vector a permutat.
- *int*:* El vector de permutacions.
- *int:* Dimensió dels vectors.

Retorna: —

4.10 double** inverse(double **a, int n)

Descripció: Inverteix la matriu factoritzant $A = LU$ i usant resolent n sistemes lineals $LUx = Id_i$, on Id_i és la i-èssima columna de la matriu identitat. Finalment, transposa la matriu.
Paràmetres:

- *double**:* La matriu a invertir.
- *int:* Dimensió de la matriu.

Retorna: La inversa de la matriu.

4.11 double** sum(double **a, double **b, int n)

Descripció: Calcula la suma $A + B$.
Paràmetres:

- *double**:* La matriu A.
- *double**:* La matriu B.
- *int:* Dimensions de les matrius.

Retorna: La suma $A + B$.

4.12 double** product(double **a, double **b, int n)

Descripció: Calcula el producte $A \cdot B$.
Paràmetres:

- *double**:* La matriu A.
- *double**:* La matriu B.
- *int:* Dimensions de les matrius.

Retorna: El producte $A \cdot B$.

5 Operacions amb polinomis / Interpolació

5.1 double horner(double z, int n, double *x, double *f)

Descripció: Evalua el polinomi en la base $(x - x_i)$, en el punt z.
Paràmetres:

- *double:* El punt en el que volem evaluar el polinomi.
- *int:* El grau del polinomi.
- *double*:* El vector de les $X = (x_i)$.
- *double*:* El vector de la evaluació $f(X) = (f(x_i))$

Retorna: El valor del polinomi en el punt z.

5.2 void hornerd(double z, int n, double *x, double *f, double p[2])

Descripció: Evalua el polinomi i la seva derivada, en la base $(x - x_i)$, en el punt z.
Paràmetres:

- *double:* El punt en el que volem evaluar el polinomi.
- *int:* El grau del polinomi.
- *double*:* El vector de les $X = (x_i)$.

– *double*:* El vector de la evaluació $f(X) = (f(x_i))$

– *double[2]:* El vector de valors de les evaluacions del polinomi i de la seva derivada en el punt z. La primera component és el valor de la evalaució del polinomi i la segona component la de la seva derivada.

5.3 void difdiv(int n, double *x, double *f)

Descripció: Calcula els coeficients del polinomi interpolador usant el mètode de les diferències dividides.
Paràmetres:

– *int:* El grau del polinomi.

– *double*:* El vector de les $X = (x_i)$.

– *double*:* El vector de la evaluació $f(X) = (f(x_i))$

5.4 void nodes(int indic, int n, double min, double max, double *x)

Descripció: Crea un vector de valors per a les $x = (x_i) \in [min, max]$, usant diferents mètodes.
Paràmetres:

– *int:* El mètode que volem usar:

1. Punts equidistants
2. Punts de Chebyshev
3. Punts aleatòris diferents.

– *int:* El grau del polinomi.

– *double:* La cota inferior dels punts.

– *double:* La cota superior dels punts.

– *double*:* El vector de les $X = (x_i)$.