Universite Batna 2
Faculté de Technologie
Département D'hydraulique

2ème Année ST_ Hydraulique
Informatique 3

A.AMEZIANE

Informatique 3

TP4: Vecteurs et matrices

Généralités

Un *tableau* (array) est un ensemble « rectangulaire » d'éléments de même type (plus précisément de même variante de type), repérés au moyen d'indices entiers.

Par opposition, on qualifiera de *scalaire* un élément du tableau. Les éléments d'un tableau sont rangés selon un ou plusieurs« axes » appelés *dimensions* du tableau.

Dans les tableaux à une dimension (qui permettent de représenter par exemple des vecteurs au sens mathématique du terme), chaque élément est repéré par un seul entier, mais le fortran accepte des tableaux jusqu'à 7 dimensions, où chaque élément est désigné par un 7-uplet d'entiers.

4. Les tableaux (array)

4.1 Quelques définitions

1. Le rang d'un tableau est son nombre de dimensions.
2. L'étendue d'un tableau dans une dimension donnée désigne le nombre d'éléments du tableau dans cette même dimension.
3. Le profil (shape) d'un tableau est un vecteur dont le $i^{ème}$ élément est le nombre d'éléments du tableau dans la $i^{ème}$ dimension.
4. Deux tableaux sont dits conformants s'ils ont le même profil.
5. La taille d'un tableau est le produit des éléments de son vecteur profil.

Exemple :

```
integer, dimension ( -5:4, 0:2) ::  x
integer, dimension (0:9, -1:1) ::  y
```

Explication :

Les tableaux *x* et y, tous deux de rang 2, ont pour profil le vecteur (/ 10, 3 /). Ils sont donc conformants.
Leur taille vaut 30.

Informatique 3 A.ameziane Année 2020/2021 Page 1

4.2 Premières caractéristiques

1. Les tableaux de rang strictement supérieur à 7 sont interdits.
2. Tout scalaire est supposé conformant à tout tableau. Par conséquent, si *a* est un tableau, l'instruction *a=1* a un sens et signifie que tous les éléments de *a* sont affectés à 1.
3. Si *a* est un tableau de dimension 2, écrire *a=1* est équivalent à écrire *a(:,:)=1*.

   De la même façon, *a(i,:)* désigne la $i^{ème}$ ligne de *a*.
4. Tout tableau de rang 1 peut être initialisé à sa déclaration en encadrant ses valeurs d'affectation par **(/ et /)**.

   Pour les tableaux de rang au moins égal à 2, on utilisera la fonction RESHAPE.

   Exemples :

   character(len=1), dimension(3) :: tabc=(/ 'a', 'b', 'c' /)

   integer, dimension (4) :: tabi1, tabi2

   tabi1 = (/ 8, 45, 6, 1 /)

   tabi2 = (/ (i*2, i=1,4) /)
5. La fonction intrinsèque **SHAPE(array)** permet d'obtenir le profil d'un tableau.
6. La fonction intrinsèque **SIZE(array [,dim])** retourne la taille d'un tableau si seul ce tableau est spécifié en argument. Si un second argument, *dim*, est précisé, cette fonction retourne l'étendue du tableau relativement à la dimension *dim*.

   Exemple :

   integer, dimension (5:4, 0:2) :: x

   print *, size(x) ! retourne 30

   print *, size(x,1) ! retourne 10

   print *, size(x,2) ! retourne 3

4.3 Sections de tableaux

Une section de tableau désigne une partie de tableau.

4.3.1 Sections régulières de tableaux :

Lorsque les indices des éléments varient en suivant une suite arithmétique, on parlera de section régulière de tableau. Pour chaque indice du tableau, cette progression arithmétique sera notée sous la forme **valeur_initiale : valeur_finale : pas**

Exemple :

integer, dimension (9,9) :: a
integer, dimension (3,3) :: b
integer, dimension (4,9) :: c
...

Informatique 3 A.ameziane Année 2020/2021 Page 2

b = a (1:5:2, 1:9:3) ! affectation possible car les tableaux sont conformants

c = a (1:7:2, :) ! affectation possible car les tableaux sont conformants

	11 12 13 14 15 16 17 18 19				
	21 22 23 24 25 26 27 28 29				
	31 32 33 34 35 36 37 38 39				
	41 42 43 44 45 46 47 48 49		11 14 17		11 12 13 14 15 16 17 18 19
Ainsi, si a vaut	51 52 53 54 55 56 57 58 59	, b vaut	31 34 37	et c vaut	31 32 33 34 35 36 37 38 39
	61 62 63 64 65 66 67 68 69		51 54 57		51 52 53 54 55 56 57 58 59
	71 72 73 74 75 76 77 78 79				71 72 73 74 75 76 77 78 79
	81 82 83 84 85 86 87 88 89				
	91 92 93 94 95 96 97 98 99				

4.3.2 Sections non régulières de tableaux :

Lorsque l'on souhaite extraire une section non régulière d'un tableau, on accède aux éléments du tableau par l'intermédiaire de vecteur d'indices.

Exemple :

```
integer, dimension (9,9) :: a
integer, dimension (10,10) :: b
...
b = 0
b ((/ 3, 5, 8 /), (/ 1, 5, 7 /)) = a (1:5:2, (/ 1, 5, 9 /))
b ((/ 1, 2 /), (/ 2, 3, 4 /)) = 1
```

	11 12 13 14 15 16 17 18 19		0 1 1 1 0 0 0 0 0 0
	21 22 23 24 25 26 27 28 29		0 1 1 1 0 0 0 0 0 0
	31 32 33 34 35 36 37 38 39		11 0 0 0 15 0 19 0 0 0
	41 42 43 44 45 46 47 48 49		0 0 0 0 0 0 0 0 0 0
Ainsi, si a vaut	51 52 53 54 55 56 57 58 59	, b vaut	31 0 0 0 35 0 39 0 0 0
	61 62 63 64 65 66 67 68 69		0 0 0 0 0 0 0 0 0 0
	71 72 73 74 75 76 77 78 79		0 0 0 0 0 0 0 0 0 0
	81 82 83 84 85 86 87 88 89		51 0 0 0 55 0 59 0 0 0
	91 92 93 94 95 96 97 98 99		0 0 0 0 0 0 0 0 0 0

Informatique 3 A.ameziane Année 2020/2021 Page 3

0 0 0 0 0 0 0 0 0 0

4.4 Opérateurs définis sur les tableaux

Les opérateurs d'affectation (=), de comparaison (==, /=, < , < =, > , > =), arithmétiques (+, , *, /, **) et logiques (.and., .or., .not.) ont été définis sur les tableaux. Ces opérateurs s'appliquent sur des tableaux de profil identique. Ces opérateurs s'appliquent terme terme.

Remarque :

Les opérateurs arithmétiques *, ** n'opèrent pas comme des opérateurs de produit matriciel sur les vecteurs et matrices mais comme des opérateurs de produit terme terme.

4.4.1 Les principales fonctions intrinsèques définies sur les tableaux

- **RESHAPE(source, shape [,pad] [,order])**

  permet de construire un tableau de profil égal à shape à partir des éléments du vecteur *source* (qui est obligatoirement un vecteur de rang 1). Si des éléments sont manquants dans *source*, le tableau est complété à l'aide des éléments figurant dans *pad*. *order* spécifie sous la forme d'un tableau de rang 1, l'ordre dans lequel les indices commencent à varier pour le remplissage du tableau résultat.

  Exemple :

  L'instruction reshape((/ (i, i=1,6) /), (/ 2, 3 /), order=(/ 2, 1 /)) permet de générer la matrice

  1 2 3

  4 5 6

- **TRANSPOSE(matrix)**,

  où *matrix* est un tableau de dimension 2, retourne la transposée de *matrix.*

- **DOT_PRODUCT(vector a, vector b)**

  retourne le produit scalaire de deux vecteurs, c'est à dire $a^t.b$ si *a* et *b* sont de type entier ou réel, conjugué$(a)^t.b$ si *a* et *b* sont complexes, et Somme(a_i .and. b_i) si *a* et *b* sont de type logique.

- **MATMUL(matrix a, matrix b)**

  retourne le produit matriciel de *a* et *b*, sous réserve de compatibilité des dimensions. Soit *a* et *b* sont tous deux des matrices, soit *a* est un vecteur et *b* est une matrice, soit *a* est une matrice et *b* est un vecteur. *a* et *b* peuvent être de type quelconque.

- LBOUND(array [,dim]) et UBOUND(array [,dim])
  retournent les bornes inférieures / supérieures de chacune des dimensions (ou seulement de la dimension *dim*) du tableau *array*.

  Exemple :

  integer, dimension(21:2, 45:49) :: tab
  lbound(tab) vaut (/ 21, 45 /) et ubound(tab) vaut (/ 2, 49 /)
  lbound(tab, dim=2) vaut 45 et ubound(tab, dim=1) vaut 2

- ALL(mask [,dim])

Informatique 3 A.ameziane Année 2020/2021 Page 4

Applique un masque de type logique sur les éléments du tableau et renvoit vrai si pour tous les éléments le résultat du masque est vrai. Si *dim* est précisé, la fonction travaille sur cet indice pour chaque valeur des autres dimensions.

Exemple :

Si A = 1 2 3 et B = 1 3 5

4 5 6 4 2 6

all(A==B) vaut .false.

all(A==B, dim=1) vaut (/ .true., .false., .false. /)

- **ANY(mask [,dim])**

Fonctionne de la même façon que la fonction ALL à l'exception qu'elle renvoit vrai si l'un des résultats du masque est vrai.

Exemple :

En reprenant A et B de l'exemple précédent, on obtient :
any(A==B) et any(A/=B) valent .true.
any(A/=B, dim=1) vaut (/ .false., .true., .true. /)

- **COUNT(mask [,dim])**

comptabilise le nombre d'éléments pour lesquels le résultat du masque est vrai.

Exemple :

count(A==B) vaut 3
count (A/=B, dim=1) vaut (/ 0, 2, 1 /)

- **MINVAL(array [,dim][,mask])** et **MAXVAL(array [,dim][,mask])**

retournent la plus petite / plus grande valeur du tableau *array* (après un éventuel filtrage par *dim* et / ou *mask*).

Exemple :

maxval(A) vaut 6
minval(A, dim=1) vaut (/ 1, 2, 3 /)
minval(A,dim=2,A > 2) vaut (/ 3, 4 /)

- **PRODUCT(array [,dim][,mask])** et **SUM(array [,dim][,mask])**

retournent le produit / la somme des valeurs des éléments du tableau *array* (après un éventuel filtrage par *dim* et / ou *mask*).

Exemple :

product(A) retourne 720
sum(A,dim=1,A > 2) vaut (/ 4, 5, 9 /)

Informatique 3 A.ameziane Année 2020/2021 Page 5

4.5 La liste complète des fonctions intrinsèques sur les tableaux.

4.5.1 L'instruction WHERE

L'instruction WHERE permet d'effectuer des opérations sur des éléments d'un tableau sélectionnés via un filtre de type logique.

Syntaxe:

```
WHERE (filtre)
  bloc1
ELSEWHERE
  bloc2
END WHERE
```

Lorsque bloc2 n'existe pas et que bloc1 se résume à une seule instruction, on peut utiliser également la forme contractée suivante :

WHERE (filtre) instruction

Exemples :

```
real, dimension(10) :: a
  ...
 where (a > 0.) a = sqrt(a)
where (a > 0.)
a = log(a)
elsewhere
a = 1.
end where
```

4.5.2 Les tableaux dynamiques

Fortran 90 permet de créer des tableaux de manière dynamique. Pour cela, il faut tout d'abord spécifier l'attribut **ALLOCATABLE** lors de la dclaration d'un tableau. Les instructions ALLOCATE et DEALLOCATE, ainsi que la fonction intrinsèque ALLOCATED permettent alors de gérer les tableaux dynamiques :

- **ALLOCATE(array, stat=err)**
  Permet de faire l'allocation mémoire du tableau. Si le mot clé stat= est spécifié, la variable *err* (de type integer) vaut 0 si l'allocation s'est déroulée sans problème et 1 sinon.
- **DEALLOCATE(array, stat=err)**
  Permet de libérer l'espace mémoire réservée à un tableau.
- **ALLOCATED(array)**
  Est une fonction intrinsèque renvoyant 0 ou 1 suivant que le tableau spécifié en argument est alloué ou non.

Informatique 3 A.ameziane Année 2020/2021 Page 6

Exemple:

```
integer, dimension (:,:), allocatable :: a
integer :: n, m, err
  ...
read *, n, m
if (.not. allocated(a)) then
allocate(a(n,m),stat=err)
if (err /= 0) print *,'Erreur d''allocation dynamique du tableau a'; stop
end if
  ...
deallocate(a)
```

Remarques:

- Un tableau ayant l'attribut ALLOCATABLE doit avoir été alloué avant d'être passé en argument d'une procédure.
- Un tableau alloué dynamiquement dans une unité de programme et n'ayant pas l'attribut SAVE (sert pour les variables locales d'une procédure. Avant chaque sortie de procédure, la valeur courante d'une telle variable est sauvegardée) a un état indéterminé en sortie de cette unité (c'est à dire après un RETURN / END). Il ne sera pas non plus libéré automatiquement.

4.6 Enoncé TP : Tableaux

Ecrire un programme qui permet d'effectuer le produit de 2 matrices A et B dont les dimensions sont stockées dans des constantes. Affiche à l'écran la matrice résultat C ligne par ligne.

- **Le programme fortran s'ecrit comme suit :**

```
program produit_mat

!------------ but du programme ----------------------------------------!
! ce programme permet d'effectuer le produit de 2 matrices
! A et B dont les dimensions sont stockées dans des
! constantes. Affiche à l'écran la matrice résultat C ligne
! par ligne.
!-------------------------------------------------------!

implicit none
! ------------ variables
integer , parameter : : m= 2 , n= 2 , p=3
integer  : :  i , j , k
integer  : :  coeff
integer  , dimension  ( m , n ) : : A
integer , dimension ( n , p ) : : B integer
, dimension ( m , p ) : : C
!-------
!--- lecture de A et B
print     ,'rentrer les éléments de A'
do i=1 , m
```


Informatique 3 A.ameziane Année 2020/2021 Page 7

```
        do j=1 , n
            print *,'A(',i,',',j,')='
            read *, A (i , j)
        end do
    end do

    print *,'rentrer les éléments de B'
    do i=1 , n
        do j=1 , p
            print *,'B(',i,',',j,')='
            read *, B (i , j)
        end do
    end do

    !--- calcul du produit
    !- boucle sur les elements de C
    do i=1 , m
        do j=1 , p
            !- calcul de C(i,j) = somme sur k des A(i,k)*B(k,j)
    coeff=0.0
            do k=1 , n
        coeff=coeff+A(i,k) B(k,j)
            end do
            C ( i , j)=coeff
        end do
    end do
    !--- affichage de C
    do i=1 , m
        print *, C ( i , : )
    end do
end program produit_mat
```

Q1 : Corrigé les erreurs de compilation
Q2 : Exécuter le programme avec votre choix
Q3 : Augmenter les dimensions est utiliser la notion de fichier pour la lecture des matrices A et B
Q4 : Ecrire un programme qui inverse les élement d'un tableau B de dimension 100.

Fin

Informatique 3 A.ameziane Année 2020/2021 Page 8