

# Initiation à Scilab

## 3 Vecteurs, matrices

Dans cette séance, l'accent est mis sur les différentes possibilités de construction et de manipulation de vecteurs et de matrices.

**Construction de vecteurs.** Différentes possibilités de construire un vecteur (appelé  $v$  par la suite) sont offertes à l'utilisateur :

- lorsque la taille du vecteur est connue et est petite, en énumérant les composantes du vecteur ; par exemple  $v=[1,2]$  pour un vecteur ligne et  $v=[1;2]$  pour un vecteur colonne,
- lorsque les valeurs du vecteur suivent une progression arithmétique, en écrivant par exemple  $v=2 :0.1 :4$ ,
- en initialisant le vecteur au vecteur nul (avec l'instruction `zeros`) ou au vecteur formé de 1 (avec l'instruction `ones`) puis en effectuant une boucle sur les indices avec des affectations du type  $v(i)=2$ ,
- en initialisant  $v$  au vecteur vide :  $v=[]$  puis en concaténant dans une boucle chaque nouvel élément avec des affectations du type  $v=[v,2]$ ,
- en effectuant des opérations de somme (instruction `+`), de multiplication (instruction `.*`) et de division (instruction `./`) terme à terme à partir de vecteurs simples.

**Exercice 3.1** Construire de trois manières différentes, le vecteur ligne de taille  $n$  qui comporte les  $n$  premiers carrés des nombres entiers.

**Construction de matrices.** Les vecteurs étant pour Scilab des cas particuliers de matrices de taille  $n \times 1$  ou  $1 \times n$ , il est naturel que la construction d'une matrice  $A$  s'effectue de manière similaire à celle d'un vecteur, en l'occurrence :

- lorsque la taille de la matrice est connue et petite, en écrivant par exemple  $A=[1,2,3 ;3,4,5]$  pour une matrice de taille  $2 \times 3$ ,
- en initialisant  $A$  à la matrice nulle (ou à la matrice identité avec `eye`) puis en effectuant une double boucle sur les indices avec des affectations du type  $A(i,j)=2$ ,
- en initialisant  $A$  à un vecteur ligne (ou colonne) puis en concaténant dans une boucle chaque nouvelle ligne (ou colonne) avec des affectations du type  $A=[A ;v]$  (respectivement  $A=[A,v]$ ). A noter que cette méthode s'étend à la concaténation entre matrices,
- en effectuant des opérations de somme (instruction `+`), de multiplication (instruction `.*`) de division (instruction `./`) terme à terme ou de multiplication matricielle (instruction `*`) à partir de matrices simples.

**Exercice 3.2** Construire de trois manières différentes, la matrice de taille  $12 \times 5$  qui comporte des zéros partout sauf sur les premières et dernières ligne et colonne qui ne comportent que des 1.

**Exercice 3.3** Construire sans effectuer de boucles la matrice  $10 \times 10$  donnant les résultats de la table de multiplication de 1 à 10.

**Extraction de sous-matrices.** Il est possible d'extraire facilement avec Scilab une sous matrice d'une matrice quelconque simplement en construisant le vecteur formé par les indices de lignes et celui formé par les indices de colonnes à sélectionner. Par exemple, l'instruction `B=A(1 :2 :5,1 :3)` extraira de la matrice A, l'intersection des lignes 1, 3 et 5 et des colonnes 1, 2 et 3 pour former une matrice de taille  $3 \times 3$ . Ainsi une solution de l'exercice 3.2 est `A=ones(12,5) ; A(2 :11,2 :4)=0`.

**Exercice 3.4** À partir de la matrice de l'exercice 3.3, construire la matrice formée de la table de multiplication des quatre premiers entiers impairs (en ligne) par les quatre premiers entiers pairs (en colonnes)

**Résolution de systèmes linéaires.** Une des premières utilisations de Scilab consiste en la résolution rapide des systèmes de  $n$  équations linéaires à  $n$  inconnues. Les instructions `det` ou `rank` permettent de savoir si un tel système possède une unique solution puis l'instruction `linsolve` (ou `\`) permet de résoudre le système.

**Exercice 3.5** Ecrire un script Scilab permettant de tester la validité de l'instruction `linsolve` avec une matrice aléatoire de grande taille et un second membre pour lequel la solution exacte est connue. On pourra utiliser avec profit l'instruction `sum` pour calculer l'erreur moyenne commise sur les coefficients du vecteur solution. Toujours pour une matrice aléatoire de grande taille, comparer, à l'aide de la commande `timer()`, les temps mis pour résoudre une système linéaire avec chacune des commandes `linsolve` et `\`.