# Maîtrise de Mathématiques Fondamentales et Applications
# Grands Systèmes linéaires
# feuille de TP 1

17 septembre 2009

---

**Exercice -** 1 *Génération des matrices particulières*

---

Le but de cet exercice est de générer aléatoirement des matrices particulières. Ces matrices seront utiles pour tester différents algorithmes. *(Les constructions proposées devront être justifiées)* .

**1-** Soit $n, m$ deux entiers naturels. Que fait la commande MATLAB suivante : `rand(n,m)` ?

**2-** Écrire une fonction de prototype `function [A] = MatSymetrique(n)`, renvoyant une matrice réelle symétrique de taille $n \times n$.

**3-** Écrire une fonction de prototype `function [A] = MatReguliere(n)`, renvoyant une matrice réelle inversible de taille $n \times n$.

**4-** Écrire une fonction de prototype `function [A] = MatReguliereI(n)`, renvoyant une matrice réelle triangulaire inférieure inversible de taille $n \times n$.

**5-** Écrire une fonction de prototype `function [A] = MatReguliereS(n)`, renvoyant une matrice réelle triangulaire supérieure inversible de taille $n \times n$.

**6-** Écrire une fonction de prototype `function [A] = MatHasard(m,n,p)`, renvoyant une matrice réelle de dimension $m \times n$, dont les éléments sont choisis aléatoirement entre les valeurs $-p$ et $p$.

**7-** Écrire une fonction de prototype `function [A] = MatHasardBin(m,n)`, renvoyant une matrice réelle de dimension $m \times n$, dont les éléments sont choisis aléatoirement parmi les deux valeurs 0 et 1.

**8-** Écrire une fonction de prototype `function [A] = MatSdp(n)`, renvoyant une matrice réelle symétrique définie positive de taille $n \times n$.

**9-** Écrire une fonction de prototype `function [A] = MatDiagDom(n)`, renvoyant une matrice réelle à diagonale dominante de taille $n \times n$.

**10-** Ecrire une fonction MATLAB de prototype `function [A]= MatRang(m,n,r)`, renvoyant une matrice réelle de taille $m \times n$ de rang $r$ fixé.

---

**Exercice -** 2 *Algorithme de Gram-Schmidt*

---

Soit $a_1, \ldots, a_n$ une famille de $n$ vecteurs de $\mathbb{R}^m$ et $A \in \mathcal{M}_{m,n}$ la matrice dont les colonnes sont formées des vecteurs $a_j$. On note $r$ le rang de la matrice $A$. On se propose d'utiliser le procedé d'orthonormalisation de Gram-Schmidt pour construire une famille orthonormale de $r$ vecteurs $u_1, \ldots u_r$ de $\mathbb{R}^m$. On considère l'algorithme suivant

Algorithme de Gram-Schmidt

```
---------------------------
Données: A. Résultats : U
---------------------------
pour p = 1 à n
  s = 0
  pour k = 1 à p-1
      s = s + (a_p, u_k) u_k
  fin
  s = a_p - s
  si || s || <> 0 alors
     u_p = s/||s||
  sinon
      u_p = 0
  fin
fin
---------------------------
```

**1-** Ecrire un fonction MATLAB de prototype `[U]=monGramSchmidt(A)`, qui implémente cet algorithme.

**2-** Tester ce programme sur quelques matrices $A \in \mathcal{M}_{m,n}(\mathbb{R})$. On calculera $UU^t$ et $U^tU$ et on commentera les résultats obtenus.

**3-** On considère l'algorithme suivant

Algorithme de Gram-Schmidt compact

```
---------------------------                  | Explication des notations
Données: A. Résultats : U                    |---------------------------------
---------------------------                  | A_k : k-ieme vecteur colonne de A
U_1 = A_1/||A_1||                            |
M   = I_m                                    | I_m : matrice identité de taile m
pour p = 2 à n                               |
  Q = U_p-1                                    |
  M = M - Q Q^t                              | Q^t : transposé de Q
  U_p = M * A_p                              |
  normUp = ||U_p||                           ||| ||: norme euclidienne
  si normUp <> 0 alors                       | <>  :  différent
    U_p = U_p/normUp                         |
  fin                                        |
end                                          |
```

**a-** Expliquer ce que fait cet algorithme.(*On remarquera qu'à l'itération p, $U_p = MA_p$, avec $M = I_m - HH^t$ ; où $H$ est la matrice $H = [U_1|\dots|U_{p-1}]$. On montrera alors que $M$ est une projection orthogonale sur le sous-espace orthogonal à $Vect(U_1, \dots U_{p-1})$*).

**b-** Implémenter cet algorithme sous MATLAB à travers une fonction de prototype `function [U] = monGramSchmidtM(A)`. Tester cette fonction sur des matrices particulières.

**c-** Comparer ce nouvel algorithme dit de "Gram-Schmidt modifié" au précédent.
(*On pourra soit compter le nombre d'opérations, soit quantifier les temps d'exécution. Dans ce dernier cas on pourra recourir aux commandes* `tic, toc, cputime` *de* MATLAB ).

**4-** Evaluer l'algorithme de Gram-Schmidt sur des matrices carrées **régulières**.
(*On utilisera la commande* `MatReguliere(n)`, $n = 3, 5, 6$ pour créer ces matrices).
Evaluer à chaque fois le produit $U^tA$ et vérifier qu'on obtient une matrice triangulaire supérieure.
Démontrer rigoureusement ce résultat.