MINF0402 TP1

Scilab - Manipulations de base

v1.2 - TU

NB le TP0 est considéré comme fini, vous êtes censés l'avoir clôturé chez vous, ce n'est pas l'objet des nouvelles séances de le terminer.
Ce TP1 s'étale sur deux séances.
Les exercices marqués "**" feront l'objet d'un rapport, les modalités vous en seront ultérieurement précisées.**

Scilab est par essence un logiciel permettant des calculs vectoriels sur les tableaux, néanmoins pour s'initier au fonctionnement et à la programmation sous Scilab, quoi de plus simple que s'entraîner à faire - "à la main" - les manipulations de tableaux les plus courantes. Les instructions dont vous avez besoin vous ont toutes été introduites dans le TP0 auquel, je vous invite à vous reporter (en particulier pour la syntaxe des boucles et des fonctions)

Les facilités vectorielles de Scilab ne seront donc pas utilisées, les calculs devront s'effectuer élément par élément ! Sauf au besoin à la console pour effectuer différents tests.

Distinguer les notions de **scripts** et les notions de **fonctions**. C'est purement conventionnel, mais on utilise le suffixe **.sci** pour les fichiers de fonctions et **.sce** pour les fichiers de scripts sous scilab
Ne pas oublier de consulter les "notes" à la fin de ce sujet

Pour mettre au point vos différents algorithmes qui doivent être écrits sous forme de fonctions, commencer par les écrire sous forme de scripts et faites les tests associés, vous êtes à la console : vous avez accès direct à toutes les variables, vous pouvez les examiner, voir les modifier.

Quand vos programmes sont au point, c'est à dire pas seulement tournent, mais donnent le résultat correct, alors passez à l'écriture sous forme de fonctions. Pour savoir si ce que vous avez écrit est correct, **on ne m'appelle pas, pas plus que votre mère : on fait des tests ! ! !**

Remarque : Quand vos fonctions sont au point - quand vous lancez l'exécution depuis l'éditeur scinotes, la fonction est simplement interprété (au sens compilé - si on veut) et cela devient une fonction scilab comme une autre que l'on peut appeler avec les arguments nécessaires à la console ou dans un script scilab.

Il est important en général, de prédéfinir les variables de sortie quand il s'agit de tableaux - si elles ne l'ont pas déjà été - en utilisant l'instruction ">zeros" voir les exemples dans les "notes" à la fin du sujet.

On peut regroupe les fonctions dans sun seul fichier, mais il faut dans ce cas lancer l'exécution de ce fichier avec la commande ">exec" (attention à la notion de repertoire courant sous scilab) le mieux est que tous les fichiers se trouvent dans le même répertoire. Une autre solution consiste à écrire les fonctions nécessaires au début du fichier de script.

Pour effectuer un affichage spécifique lors de l'exécution d'un script ou d'une fonction on peut utiliser l'instruction ">disp", pour faire des pauses à l'affichage on peut par exemple utiliser :l'instruction >sleep(4000) pour des pauses de 4 secondes. Pour les scripts pour avoir des affichages supplémentaires à l'écran on peut les lancer avec l'instruction ">exec" et spécifier le paramètre de "mode" supplémentaire (voir le help).

Faites des tests avec des matrices d'ordre supérieur ou égale à 5 générées automatiquement, surtout pas de script de saisie à la main pour les matrices ! ! , fermer ou détacher au besoin l'explorateur de fichier, ou l'explorateur de variables dans la fenêtre Scilab de façon à avoir plus d'espace pour la console d'exécution ce qui permet d'afficher des matrices 5x5 sans retour à la ligne, dans le TP0, vous avez vu en particulier comment générer des matrices aléatoires, triangulaires, diagonales etc ...

Les noms des fonctions sont imposés, vous devez les respecter.

Exercice 1 (basique)

A et B désignant deux tableaux mono-dimensionnels de flottants quelconques de longueur n, construire la fonction MON_SCA qui sur la donnée de A, B et n **- variable d'entrée -** renvoie le produit scalaire de A et B dans SCA **- variable de sortie -** .

Exercice 2 (basique)

A, B, C et D désignant quatre tableaux bidimensionnels de type (n,m), construire les fonctions suivantes :

1. Une fonction SUMB qui sur la donnée de A, B, n et m **- variables d'entrée -** renvoie la somme $A + B$ dans C **- variable de sortie -** .
2. Une fonction SUMDIFF qui sur la donnée de A, B, n et m **- variable d'entrée -** renvoie $A+B$ et $A - B$ dans C et D **- variables de sortie -** . (Attention il y a une petite subtilité dans l'appel de la fonction)

Exercice 3 (basique)

A désignant un tableau de type (n,m) et B un tableau de type (m,q). Construire la fonction suivante :

* Une fonction PRODMAT qui sur la donnée de A, B, n, m et q **- variables d'entrée -** renvoie le produit matriciel $A * B$ dans un tableau C de type (n,q) **- variable de sortie -** .

Exercice 4 (**)**

A désignant un tableau de type (n,n) correspondant à une matrice inversible et b un vecteur colonne à n composantes. Construire les fonctions suivantes :

1. Une fonction RESOUINF qui sur la donnée de A et b et n **- variables d'entrée -** renvoie la solution X **- variable de sortie -** de $Ax = b$, lorsque A correspond à une matrice triangulaire inférieure inversible.
2. Une fonction RESOUSUP qui sur la donnée de A et b et n **- variables d'entrée -** renvoie X **- variable de sortie -** la solution de $Ax = b$, lorsque A correspond à une matrice triangulaire supérieure inversible. matrice triangulaire.

On devra bien entendu tester le fonctionnement correct de ces deux fonctions sur des exemples avec des matrices au moins 5×5 et dans le cas où l'on connait directement la solution (On peut se donner A triangulaire et x en déduire b et regarder si la méthode de résolution nous redonne bien b. (N.B. pour une matrice triangulaire, la matrice sera inversible ssi tous les coefficients diagonaux sont non nuls)

Exercice 5 (**)**

A désignant un tableau de type (n,n) correspondant à une matrice inversible et b un vecteur colonne à n composantes. On suppose de plus que la méthode de Gauss est possible sur A sans permutation de lignes (on ne rencontre pas de pivot nul) Construire les fonctions suivantes :

1. Une fonction REDUC (pour *réduction de Gauss*) qui sur la donnée de A et b et n **- variables d'entrée -** renvoie alors A et b **- variables de sortie -** où A est alors une matrice triangulaire supérieure et b un vecteur (colonne) à n composantes, et où $Ax = b$ est le nouveau système triangulaire supérieur obtenu à l'issue de la réduction de Gauss. On pourrait en fait travailler avec des variables de sortie distinctes mais c'est beaucoup plus naturel ainsi. A noter qu'il sera encore possible d'utiliser des variables de sortie distinctes lors de l'appel à la fonction.
2. Une fonction GAUSS qui sur la donnée de A et b et n **- variables d'entrée -** renvoie X **- variable de sortie -** la solution de $Ax = b$. On utilisera la fonction REDUC ci-dessus et la fonction RESOUSUP de l'exercice précédent. Il n'y a presque rien à écrire pour cette fonction...

On devra bien entendu tester le fonctionnement correct de ces deux fonctions. Il pourra être bon temporairement dans REDUC de visualiser la forme correcte de la matrice, après chaque itération de Gauss. (NB un matrice carrée quelconque prise au hasard (véritablement au hasard, c'est à dire pas prise au hasard "à la main" sera presque surement toujours inversible -))

Exercice 6 (**)**

On reprend le dernier exercice du TD2 dont on rappelle ici l'énoncé :

Exercice 6 (TD2)

On veut résoudre le système linéaire $Mx = d$ de la forme suivante :

$$\begin{pmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & a_4 & b_4 & c_4 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \dots \\ d_{n-1} \\ d_n \end{pmatrix}$$

On a donc ici une matrice tridiagonale. On suppose donc que M correspond à une matrice inversible et que l'on peut effectuer la méthode de Gauss systématique sans permutation de ligne (ou colonnes).

1. Montrer que l'on peut se ramener à un système de la forme :

$$\begin{cases} x_i = e_i x_{i+1} + f_i & 1 \leq i \leq n-1 \\ x_n = f_n \end{cases}$$

   Où l'on donnera les expressions récurentes définissant les e_i, $(1 \leq i \leq n-1)$ et les f_i, $(1 \leq i \leq n)$.

2. Donner l'algorithme de la méthode.

La matrice M dans ce type de problème est supposée pouvoir être de très grande taille, pour économiser de l'espace mémoire on ne stocke pas directement la matrice M mais uniquement des tableaux mono-dimensionnels A, B, C, D et X de longueur n. ([1]) (Ici pour nos test, on se contentera de petite taille : n entre 5 et 10 par exemple mais on reviendra au besoin la dessus dans le prochain TP avec des valeurs de n bien plus grandes)

On devra donc résoudre en utilisant l'algorithme vu en cours/TD ce qui nous amènera à introduire de plus des tableaux E et F mono-dimensionnels de longueur n ([2]). On devra donc construire une fonction scilab $[\texttt{X}] = \texttt{RESOUTRI}(\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}, \texttt{n})$ effectuant cette résolution.

On devra ensuite vérifier que la solution trouvée est correcte en effectuant la multiplication MX (sans former pour autant la matrice M explicitement, il faudra donc construire de plus une fonction spécifique pour effectuer le produit matriciel directement à partir de A, B, C, D et X. On calculera alors la norme de $MX - D$ qui devra donc être très petite. (pour ces tests, afin d'éviter tout problème, on prendra $|b_i| > |a_i| + |c_i|$ pour tout i)

(Il pourra bien entendu être utile, pour la mise en place/au point du programme et les tests intermédiaires de cependant former la matrice M à partir de A,B et C, cela permettra en particulier

1. La matrice M pourrait être de très grande taille car n pourrait être très grand en pratique : à 8 octets par flottant (usuel en 64bits) et $n = 10^6$ le stockage de la matrice demanderait 8×10^{12} octets soit $8 \times 10^6 Mo$ contre 24×10^6 octets ($24Mo$) pour le stockage direct des trois diagonales.

2. Formellement on pourrait prendre certains tableaux de longueur $(n-1)$, mais cela serait une optimisation sans intérêt, bien au détriment de la lisibilité

de s'assurer que la fonction de multiplication écrite ci dessus est correcte - mais cela n'aura pas à figurer dans le rapport)

——— • • ———

Notes

- Scilab peut déterminer directement la solution de $Ax = b$ pour A inversible avec l'instruction ">A\b" la division à gauche comme il a été dit dans le TP0. Ce qui vous permet d'effectuer des vérifications basique à la console, mais ce n'est pas la seule façon et ce n'est pas la méthode souhaitable pour le rapport pour des raisons évidentes. On peut avec la solution trouvée calculer $Ax - b$ voir plus bas ... On peut aussi se donner directement x calculer $b = Ax$ et chercher à résoudre avec le programme écrit, ce qui doit nous redonner le x choisi ...
- **Norme :** Si U est un vecteur de n composantes, on définit la norme (usuelle) de U par : $||U|| = \sqrt{\sum_{i=1}^{n} x_i^2}$. On obtient la norme d'un vecteur U sous Scilab avec la fonction >norm(U).
- Si x est la solution exacte de $Ax = b$ alors $Ax - b$ devrait être le vecteur nul de norme nulle. Cependant le calcul s'effectuant en flottant avec une précision limitée, même avec un calcul correct, cette quantité ne sera usuellement pas le vecteur nul, mais l'erreur $||Ax - b||$ sera (devra être) "petite", on pourra aussi introduire - pour b non nul - l'erreur relative $||Ax-b||/||b||$ exprimable alors en pourcentage.
- L'instruction ">zeros"

```
--> A=zeros(2,3)
 A  =
    0.    0.    0.
    0.    0.    0.

--> A=rand(2,3)
 A  =
    0.7263507    0.5442573    0.2312237
    0.1985144    0.2320748    0.2164633

--> B=zeros(A)
 B  =
    0.    0.    0.
    0.    0.    0.

--> A=zeros(5)    // pourquoi ?
 A  =
    0.

--> A=zeros(1,5)
 A  =
    0.    0.    0.    0.    0.
```

(Remarque l'instruction ">ones" fonctionne de façon similaire

- Revoir les instructions ">rand" ">triu" et ">tril" du TP0, pour la génération automatique de vos matrices.

——— • **FIN** • ———