MINF0402 TP1

Scilab - Manipulations de base

v1.00 - RS

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 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 le premier exercice, où on parle de **script** et les suivants où on parle de **fonction**. Pour mettre au point vos différents algorithmes qui doivent être écrits sous forme de fonctions, commencer par les écrire sous forme de scripts. Quand ceux-ci 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 ! ! !**

(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)

Exercice 1 (basique)

A et B désignant deux tableaux mono-dimensionnels de longueur n, construire les scripts suivants :

1. Un script SCRIPT_SUM qui effectue la somme de A et B renvoie la somme $A + B$ dans une variable C.
2. Un script SCRIPT_MON_SCA qui renvoie le produit scalaire SCA de A et B. (mathématiquement $\sum_{i=1}^{n} a_i b_i$)

Exercice 2 (basique)

Adum et *Bdum* désignant deux tableaux mono-dimensionnels quelconques de longueur *ndum*, construire les fonctions suivantes :

1. Une fonction MA_SOMME qui sur la donnée de *Adum*, *Bdum* et *ndum* **- variable(s) d'entrée -** renvoie la somme $Adum + Bdum$ dans *Cdum* **- variable(s) de sortie -** .
2. Une fonction MON_SCA qui sur la donnée de *Adum*, *Bdum* et *ndum* **- variable(s) d'entrée -** renvoie le produit scalaire de *Adum* et *Bdum* dans *SCAdum* **- variable(s) de sortie -** .
3. Une fonction SUM_ET_DIF qui sur la donnée de *Adum*, *Bdum* et *ndum* **- variable(s) d'entrée -** renvoie $Adum + Bdum$ et $Adum - Bdum$ dans des tableaux de même type *Cdum* et *Ddum* **- variable(s) de sortie -** .

Exercice 3 (basique)

Adum, *Bdum*, *Cdum* et *Ddum* désignant quatre tableaux bidimensionnels de type (*ndum*,*mdum*), construire les fonctions suivantes :

1. Une fonction SUMB qui sur la donnée de *Adum*, *Bdum*, *ndum* et *mdum* **- variable(s) d'entrée -** renvoie la somme $Adum + Bdum$ dans *Cdum* **- variable(s) de sortie -** .
2. Une fonction SUMDIFF qui sur la donnée de *Adum*, *Bdum*, *ndum* et *mdum* **- variable(s) d'entrée -** renvoie $Adum + Bdum$ et $Adum - Bdum$ dans *Cdum* et *Ddum* **- variable(s) de sortie -** .

Exercice 4 (basique)

Adum désignant un tableau de type (*ndum*,*mdum*) et *Bdum* un tableau de type (*mdum*,*qdum*). Construire la fonction suivante :

* Une fonction PRODA qui sur la donnée de *Adum*, *Bdum*, *ndum*, *mdum* et *qdum* **- variable(s) d'entrée -** renvoie le produit matriciel $Adum * Bdum$ dans un tableau de type (*ndum*,*qdum*) *Cdum* **- variable(s) de sortie -** .

Pour les exercices suivants (même si cela est aussi valable pour les précédents), on vous recommande de commencer par écrire un script effectuant ce qui est demandé - pas une fonction ! ! ! - testez votre script et ensuite convertissez le en fonction Scilab. L'avantage d'être dans un script c'est que vous avez accès à toutes vos variables que vous pouvez examiner à la console et que vous pouvez aussi modifier si besoin est.

Exercice 5 (**)**

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 **- variable(s) d'entrée -** renvoie la solution X **- variable(s) 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 **- variable(s) d'entrée -** renvoie X **- variable(s) 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. (Générer A de façon automatique, il pourra être pratique pour se faire d'utiliser *rand*, et *tril* ou *triu*. (N.B. pour une matrice triangulaire, la matrice sera inversible ssi tous les coefficients diagonaux sont non nuls)

Exercice 6 (**)**

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 **- variable(s) d'entrée -** renvoie alors U et y **- variable(s) de sortie -** où U est une matrice triangulaire supérieure et y un vecteur (colonne) à n composantes, et où $Ux = y$ est le système triangulaire supérieur obtenu à l'issue de la réduction de Gauss. Le plus simple est en fait de renvoyer directement U et y dans les variables A et b, ces variables doivent alors figurer dans la liste d'entrée et dans la liste de sortie
2. Une fonction GAUSS qui sur la donnée de A et b et n **- variable(s) d'entrée -** renvoie X **- variable(s) 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 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 et x en déduire b et regarder si la méthode de résolution nous redonne bien b. (Générer A de façon automatique, il pourra être pratique pour se faire d'utiliser *rand*, et *tril* ou *triu*. (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 - mais si on a un doute on peut toujours calculer le déterminant avec det(A) et la matrice sera inversible ssi son déterminant est non nul)

Exercice 7 (**)**
On reprend le dernier exercice du TD2 dont on rappelle ici l'énoncé :

> **Exercice 5 (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 :
>
> $$\left\{ \begin{array}{rcll} x_i & = & e_i x_{i+1} + f_i & 1 \leq i \leq n-1 \\ x_n & = & f_n & \end{array} \right.$$
>
> 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 `[X] = RESOUTRI(A,B,C,D,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 être utile, pour la mise en place du programme et les tests intermédiaires de cependant former la matrice M à partir de A,B et C, cela permettra en particulier que la fonction de multiplication écrite ci dessus est correcte)

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,

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

mais ce n'est pas la seule façon, 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 l'erreur relative $||Ax - b||/||b||$ exprimable alors en pourcentage.

——— • **FIN** • ———