Module: Programmation avec Mat-lab **TP 4 : Éléments de programmation**

Centre universitaire de Mila

Master1: MAF-2023/2024

Exercice 1

La fonction *produit* permet de calculer le produit de deux scalaires *x* et *y*. Si *x* ou *y* n'est un scalaire, la fonction *produit* affiche le message '*Erreur, les paramètres doivent être des scalaires*'.

1) Définir la fonction *produit* dans un m-file (nommé *produit.m*). Pour réaliser le cas d'erreur, il faut utiliser la commande: *error*.

2) Appeler la fonction *produit* à partir de la fenêtre de commande.

3) Modifier la fonction produit en remplaçant la commande *error* par la commande *disp*, puis expliquer la différence entre l'utilisation de ces deux commandes. Afin de mieux comprendre la différence entre ces deux commande, ajouter la commande *disp('hello')* après chacune de +++6+ces commandes.

4) a) Définir la fonction *produitM* qui permet de calculer le produit matriciel de deux matrice *A* et *B* quelconque (*A*B*). Si le produit n'est pas possible *produitM* affiche un message d'erreur en utilisant la commande *error*. La fonction *produitM* doit être applicable sur les vecteurs et les scalaires. **b)** tester *produitM* sur des exemples.

Exercice 2

Définir les fonctions suivantes, puis tester les sur des exemples (scalaires, vecteurs et matrices):

1) *f* : définie comme suit: $f(x) = \frac{x^5 - 3}{\sqrt{x^2 + 1}}$

2) *pair* : teste si un nombre est pair ou non.

3) *impair* : teste si un nombre est impair.

4) **a)** *puiss4* : qui élève son argument à la puissance quatre x^4. La fonction *puiss4* doit appeler la fonction *puiss2* définie dans le même fichier *puiss4.m*.

a) Est-ce qu'il est possible d'appeler *puiss2* directement à partir de la fenêtre de commande.

5) *seasons(m)* **:** qui affiche: (il faut utiliser les *if-else* imbriquées, puis la structure *swith-case*).

- *'summer'*, si *m* = 6, 7 ou 8.
- *'autumn'*, si *m* = 9, 10 ou 11.
- '*winter'*, si *m* = 1, 2 ou 12.
- *'spring*', si i = 3, 4 ou 5.
- *'erreur'* si *m* <1, ou *m*>12.

Exercice 3 (fonctions, scalaires, boucles)

Définir les fonctions suivantes, puis tester les sur des exemples (scalaires, vecteurs et matrices):

1) *fact* qui calcule *n!*, en utilisant la boucle *for*. Si n<0, la fonction affiche *'erreur'*.

2) *fact* qui calcule *n!*, en utilisant la boucle *while*. Si n<0, la fonction affiche *'erreur'*.

3) *cmb (combinaison) définie comme suit:*

$$C_n^p = \frac{n!}{p!(n-p)!}$$

Exercice 4

Définir les fonctions suivantes, puis tester les sur des exemples:

1) *estId* : qui teste si deux matrices *A* et *B* sont identiques (c'est-à-dire ayant exactement les mêmes valeurs). Si *A* et *B* sont identiques retourne *true* (1) sinon retourne *false* (0).

2) *som* : qui calcule la somme des éléments d'une matrice *A*. Il ne faut pas utiliser la fonction *sum.*

3) *Max* : qui calcule le max des éléments d'une matrice. Il ne faut pas utiliser la fonction *max.*

4) *addM*: qui calcule la somme de deux matrices: *A+B*. Il ne faut pas utiliser directement l'opérateur +. Il faut parcourir les éléments des deux matrices et faire l'addition élément par élément. Si l'addition n'est pas possible, la fonction affiche *'erreur'*.

5) *MyEye(n)*: qui permet de construire la matrice identité de dimension *n*.

6) myTril (et *myTriu*): qui permet d'extraire la partie triangulaire *inf* et (*sup*) d'une matrice.

Exercice 5

Définir les fonctions suivantes, puis tester les sur des exemples:

$$A = \begin{pmatrix} 3 & -5 & 6 \\ 2 & -4 & 0 \\ 2 & 5 & 1 \end{pmatrix}$$

$$V1 = \begin{pmatrix} 2 & 5 & 1 \end{pmatrix}$$

$$V2 = \begin{pmatrix} -5 \\ -4 \\ 5 \end{pmatrix}$$

1) *ligne(M, k)* : qui retourne un vecteur ligne contenant les élément de la ligne numéro *k* de la matrice *M*. Par exemple *V1 = ligne(A, 3)*.

2) *colonne(M, k)* : qui retourne un vecteur colonne contenant les élément de la colonne numéro *k* de la matrice *M*. Par exemple *V2 = colonne(A, 2)*.

3) *prodVect(M1, M2)* : qui calcule le produit *M1*M2*. (*M1* et *M2* sont respectivement un vecteur-ligne et un vecteur-colonne. Ils ont le même nombre d'élément. Il ne faut pas utiliser directement l'opérateur *, mais il faut parcourir les éléments des deux vecteurs et faire la multiplication. Par exemple: *prodVect(V1, V2) = 2×(-5)+5×(-4)+1×5 = -25.*

4) *prodMat*: qui calcule le produit de deux matrices: *A*B*. Il ne faut pas utiliser directement l'opérateur *, mais il faut parcourir les éléments des deux matrices et faire la multiplication en utilisant les trois fonctons *ligne*, *colonne* et *prodVect*. Si la multiplication n'est pas possible, la fonction affiche *'erreur'*.

HW : Résolution d'un système d'équations linéaires

1) Ecrire une fonction appelée ***equLin*** qui permet de :

- **a)** Demander à l'utilisateur d'entrer une matrice carrée (la matrice des coefficients A).
- **b)** Si la matrice n'est pas carrée, on affiche un message d'erreur et on termine l'exécution, sinon on continue.
- **c)** Demander à l'utilisateur d'entrer un vecteur colonne (B) dont le nombre de lignes est égale au nombre de colonnes de la matrice des coefficients A.
- **d)** Si le nombre de lignes de B est différent du nombre de colonnes de A, on affiche un message d'erreur et on termine l'exécution, sinon on continue.
- **e)** Résoudre le système d'équation linéaire AX = B, puis affiche le résultat.

2) Ecrire une autre fonction appelée ***equLin2*** qui fait la même chose que la fonction précédente, mais elle doit retourner le résultat au lieu de l'afficher. En plus, la valeur de A et B doivent être passées en permettre.