UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

UNIVERSITÉ PARIS DESCARTES

Corrigé de l'examen partiel du 23 mars 2009
L2 : Environnement de calcul scientifique et modélisation

Questions de cours :

1. Déterminons le nombre de multiplications de réels nécessaires pour obtenir la matrice `A*B`. Cette matrice est de taille $[n,p]$ et contient donc np éléments, calculés grâce à $(A*B)_{ij} = \sum_{k=1}^{m}(A)_{ik}(B)_{kj}$. Pour chacun de ces éléments, on doit effectuer m multiplications. En tout le produit matriciel `A*B` nécessite donc exactement nmp multiplications de nombres réels.

2.

a) Les dimensions de x, y et z étant respectivement $[n,1]$, $[1,n]$ et $[n,1]$, la multiplication matricielle est définie et le résultat $x*y*z$ est une matrice de taille $[n,1]$.

b) Grâce à **1**, le produit $a=(x*y)$ nécessite $n\cdot 1\cdot n = n^2$ multiplications. De même, comme a est de taille $[n,n]$, pour le produit $a*z=(x*y)*z$ on doit effectuer encore n^2 multiplications. En tout, le calcul de $(x*y)*z$ nécessite donc $2n^2$ multiplications.

c) De même, $b=(y*z)$ est de taille $[1,1]$ et son calcul nécessite n multiplications. Pour calculer $x*b=x*(y*z)$ on a besoin de n multiplications.
En tout, il faut $2n$ multiplications pour déterminer $x*(y*z)$.

d) $n=10^6$, $(x*y)*z$ nécessite $2\cdot 10^{12}$ multiplications, contre $2\cdot 10^6$ pour $x*(y*z)$. En conclusion, pour calculer le résultat $x*y*z$, il est (10^6 fois !) plus rapide de faire les calculs dans l'ordre donné par les parenthèses en **c** que celui donné en **b**.

Exercice 1

1. $A=\begin{pmatrix}2&4&6\\2&4&6\end{pmatrix}$, en inversant la seconde ligne et en prenant la transposée $B=\begin{pmatrix}2&6\\4&4\\6&2\end{pmatrix}$.

2. On définit le polynôme P grâce à ses racines, $P=2(x-1)(x-2)(x-1)=-4+10*x-8*x^2+2*x^3$. Le polynôme Q est défini à partir de ses coefficients, $Q=2(1+2*x+x^2)=2+4*x+2*x^2$, d'où : `roots(P)` donne $1,1,2$; `coeff(P)` affiche -4, 10, -8 et 2 ; `roots(Q)` donne $-1,-1$; `coeff(Q)` affiche 2, 4 et 2 ; `P-Q*poly([0],'x')` $=-4+8*x-12*x^2$.

3. `matrix(1:6,3,2)'` produit la matrice $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$. Il faut prendre $m=2$ et $n=2$ pour obtenir la matrice $C=\begin{pmatrix}1&2&3&1&0\\4&5&6&0&1\\0&0&1&1&1\\0&0&1&1&1\end{pmatrix}$

4. On a $D=\begin{pmatrix}1+i&2+2i&3+3i\\4+3i&5+2i&6+i\end{pmatrix}$. Le test `real(D)==imag(D)` compare, élément par élément, la matrice contenant la partie réelle de D avec la matrice contenant la partie imaginaire de D. Le résultat est la matrice booléenne $R=\begin{pmatrix}T&T&T\\F&F&F\end{pmatrix}$.

Exercice 2

1. Avec deux boucles `for` imbriquées :

```
A=zeros(n,n);
for i=1:n
for j=1:n
  A(i,j) = 2^(i+j)*j^2;
end
end
```

2. Pour construire la matrice $A = ((a_{ij}))_{1 \leq i,j \leq n}$ sans boucle, on peut remarquer que $a_{ij} = b_i c_j$ avec $b_i = 2^i$ et $c_j = 2^j j^2$, donc A est le produit matriciel du vecteur colonne (b_i) par le vecteur ligne (c_j) : `A= 2.^([1:n]') * (  2.^[1:n].*([1:n].^2) )`

Exercice 3

Pour évaluer `(a-b)-c`, Scilab calcule d'abord $a - b$ qui donne exactement 0, puis $0 - c$, ce qui donne exactement 10^{-17} et produit l'affichage `- 1.000E-17`.
En revanche, pour évaluer `a-(b+c)`, Scilab calcule d'abord $b + c$, c'est-à-dire $1 + 10^{-17}$. Or par définition `1+%eps` est le plus petit nombre Scilab strictement supérieur à 1, donc $1 + 10^{-17}$ n'est pas représentable par Scilab et est tronqué à 1. C'est donc à cause de la précision machine limitée que l'évaluation de `a-(b+c)` par Scilab produit l'approximation 0 (1-1) et non le résultat exact -10^{-17}.

Exercice 4

1. L'opérateur `.^` désigne le carré élément par élément donc le résultat affiché sera la matrice $A = \begin{pmatrix} 1^2 & 2^2 \\ 1^2 & 1^2 \end{pmatrix} = \begin{pmatrix} 1 & 4 \\ 1 & 1 \end{pmatrix}$.

2. L'opérateur `^` désigne le carré au sens matriciel, le résultat affiché sera $B = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 2 & 3 \end{pmatrix}$.

3. La première partie de la commande définit $x = \sqrt{2}$ (non affiché), et la deuxième partie conduit à l'affichage de la matrice $C = \begin{pmatrix} \sqrt{2} & \sqrt{2} \\ \sqrt{2}/2 & \sqrt{2} \end{pmatrix} \begin{pmatrix} \sqrt{2} & \sqrt{2} \\ \sqrt{2}/2 & \sqrt{2} \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 2 & 3 \end{pmatrix}$.

4. La double commande `diag(diag(...))` appliquée à une matrice carrée a pour effet de mettre à zéro les coefficients non diagonaux de cette matrice (le premier appel `diag()` extrait dans un vecteur les coefficients diagonaux de la matrice, et le deuxième appel `diag()` place ces coefficients sur la diagonale d'une matrice nulle). La double commande `diag(diag(...))` appliquée au résultat de `matrix(1:15,5,5)`, qui construit la matrice (non affichée) $\begin{pmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix}$, produit donc l'affichage de $D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 13 & 0 & 0 \\ 0 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 25 \end{pmatrix}$.

5. Le produit matriciel construit la matrice $E = \begin{pmatrix} 2 & 3 & 4 \\ 4 & 6 & 8 \\ 6 & 9 & 12 \end{pmatrix}$, qui n'est pas affichée à cause du point-virgule. Les commandes `E>5` et `E<9` effectuent ensuite des tests élément par élément et donnent respectivement $E > 5 = \begin{pmatrix} F & F & F \\ F & T & T \\ T & T & T \end{pmatrix}$ et $E < 9 = \begin{pmatrix} T & T & T \\ T & T & T \\ T & F & F \end{pmatrix}$.
L'opérateur booléen "et", `&`, s'applique enfin élément par élément et affiche $\begin{pmatrix} F & F & F \\ F & T & T \\ T & F & F \end{pmatrix}$.

6. Le vecteur x contient au départ les entiers de 1 à 100, avec $x(k) = k$. Pour chaque valeur de $i = 2, 3, ..10$ on met à 0 les coefficients de x dont les indices sont $k = 2i, 3i, 4i...$ c'est-à-dire les multiples non-triviaux de i contenus dans x (non-triviaux signifiant ici : i non compris). A l'issue de la boucle, les seuls $x(k)$ qui ne sont pas nuls sont les nombres entre 1 et 100 qui ne sont multiples non triviaux d'aucun entier entre 2 et 10, c'est-à-dire 1,2,3,5,7,11,13,...97. La dernière commande `x(x>1)` affiche les éléments de `x` qui sont strictement supérieurs à 1, c'est-à-dire exactement les nombres premiers inférieurs à 100.