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

UNIVERSITÉ PARIS DESCARTES
UFR DE MATHÉMATIQUES ET INFORMATIQUE

Licence 2e année, 2007–2008

Environnement de calcul scientifique et modélisation

Examen du 20 mai 2008. Durée 1h30.
Nombre de pages de l'énoncé : 3

Tout document ainsi que l'utilisation de tout appareil électronique, même à titre d'horloge, est interdit.

Justifiez vos réponses ! Il sera tenu compte de la présentation !

Questions de cours

1. Citer trois points importants pour obtenir une représentation graphique convenable d'une fonction réelle.
   On pourra prendre en exemple $f(x) = 1/x$. Expliquer clairement l'importance de chacun des points cités.
2. Comment définit-on un polynôme en Scilab lorsque l'on connaît ses coefficients ? Comment définit-on un polynôme en Scilab lorsque l'on connaît ses racines ? Quelle fonction Scilab et de quels arguments permet d'évaluer un polynôme en un point réel ?

Exercice 1.

Pour p et q deux entiers non nuls (non fixés), on désigne par A une matrice de taille (p, p), B une matrice de taille (p, q) et C une matrice de taille (q, q). On définit la matrice M par blocs : $M = \begin{bmatrix} A & B \\ O & C \end{bmatrix}$, où O est un bloc de zéros.

1. Quelle est la taille du bloc O de M ?
2. Si A, B, C sont définies dans Scilab, donner une ligne de commande qui permet de définir M.
3. On suppose désormais A et C inversibles. On pose $N = \begin{bmatrix} A^{-1} & -A^{-1}BC^{-1} \\ O & C^{-1} \end{bmatrix}$. Calculer MN et NM à la main.
4. On rappelle que `inv(P)` désigne l'inverse d'une matrice `P`. Utilisant la question précédente qui a montré comment calculer un inverse par blocs, écrire une fonction Scilab, `[ N ] = inv_blocs(A,B,C)`, qui détermine l'inverse N de la matrice M définie par les blocs A, B et C.
5. On suppose A^{-1} et C^{-1} déjà déterminées. Combien d'opérations nécessite alors le calcul de $A^{-1}BC^{-1}$
   – dans le cas où on calcule $A^{-1}B$, puis $(A^{-1}B)\,C^{-1}$,
   – dans le cas où on calcule BC^{-1}, puis $A^{-1}\,(BC^{-1})$?

Exercice 2. Recopier le numéro du sujet et reporter pour chaque question les lettres des réponses exactes sur votre copie.

Il peut y avoir plusieurs réponses exactes pour chaque question, ainsi qu'aucune réponse exacte.

1. Les commandes suivantes permettent de représenter les deux fonctions $\sin(x)$ et $\cos(x)$ sur l'intervalle $[0, \pi]$ de manière satisfaisante :

   A : `x=0:%pi;y=cos(x);z=sin(x);plot2d([x,x],[y,z],style=[5,5]);`

   B : `x=linspace(0,%pi,100);y=cos(x);z= sin(x);plot2d([x,x],[y,z],style=[5,5]);`

   C : `y=cos(x);z=sin(x);x=linspace(0,%pi,100);plot2d([x,x],[y,z],style=[5,5]);`

   D : `x=linspace(0,%pi,100);y=cos(x);z=sin(x);plot2d([x',x'], [y',z'], rect=[0,-1,%pi,1])`(dans cet énoncé, on n'est passé "à la ligne" que par manque de place sur une même ligne).

   E : `x=linspace(0,%pi,100);y=cos(x);z=sin(x);plot2d([x',x'],[y',z'],style=[5,5]);`

2. Les commandes suivantes permettent de définir le polynôme $P(x) = (x-1)(x-2)$:

   A : `P=(x-1)*(x-2)`

   B : `p1=poly(1,"x"); p2=poly([-2, 1],"x","c");P=p1*p2`

   C : `x=poly(0,"x");P=(x-1)*(x-2)`

   D : `P=poly([1,2],"x")`

   E : `p1=poly(1,"x"); p2=poly([1,-2],"x","c");P=p1*p2`

3. Les opérations suivantes définissent la fonction `pentes` qui, à une fonction `f` et à un vecteur ligne `x` de taille `n` tel que pour tout `i=2,..,n, x(i)` $\neq$ `x(1)`, associe les pentes successives de `x(1)` à `x(i), i=2,...,n`, c'est-à-dire les valeurs `(f(x(i))-f(x(1)))/(x(i)-x(1)), i=2,...,n`.

A : Sauver dans le fichier `pentes.sci` les lignes
```
function g=pentes(f,x)
n=size(x,"c");
for i=1:n-1, g(i)=(f(x(i+1))-f(x(1)))/(x(i+1)-x(1)); end;
endfunction
```
puis charger la fonction par `getf("pentes.sci")`.

B : Sauver dans le fichier `pentes.sci` les lignes
```
function g=pentes(f,x)
n=size(x,"c");
for i=1:n-1, h(i)=f(x(i+1))-f(x(1)), y(i)=x(i+1)-x(1); end;
g=h/y;
endfunction
```
puis charger la fonction par `getf("pentes.sci")`.

C : Sauver dans le fichier `pentes.sci` les lignes
```
function g=pentes(f,x)
n=size(x,"c");
for i=1:n-1, h(i)=f(x(i+1))-f(x(1)), y(i)=x(i+1)-x(1); end;
g=h./y;
endfunction
```
puis charger la fonction par `getf("pentes.sci")`.

D : Sauver dans le fichier `pentes.sce` la ligne
```
deff("g=pentes(f,x)", "g(i)=(f(x(i+1))-f(x(1)))/(x(i+1)-x(1))")
```
puis charger la fonction par `getf("pentes.sce")`.

E : Le nombre `n` étant supposé égal à `3`, écrire dans la fenêtre Scilab
```
deff("g=pentes(f,x)",
 "g(1)=(f(x(2))-f(x(1)))/(x(2)-x(1)), g(2)=(f(x(3))-f(x(1)))/(x(3)-x(1))")
```
(dans cet énoncé, on n'est passé "à la ligne" que par manque de place sur une même ligne).