Introduction au calcul scientifique (J. Ponce)

TP 1 – Introduction à Scilab

Romain Brette
brette@di.ens.fr

Scilab est un logiciel de calcul numérique axé sur la manipulation de matrices. Il est gratuit (développé par l'INRIA et l'ENPC) et disponible à l'adresse suivante: `http://www.scilab.org`.

Créez un répertoire de travail et lancez Scilab en tapant dans une console les lignes suivantes :

`mkdir scilab`	crée le répertoire (make directory)
`cd scilab`	se déplace dans le répertoire (change directory)
`scilab`	lance scilab

Manipulation de vecteurs et de matrices

Taper dans Scilab les commandes indiquées après la flèche `-->`.

`--> 5`	
`--> (3.4+9/5)*3`	
`--> abs(5)+cos(3)`	valeur absolue, cosinus
`--> %pi`	%pi est la constante π
`--> 5^2`	puissance
`--> ans + 1`	ans = dernier résultat (answer)
`--> sqrt(16)`	racine carrée (square root)
`--> %i^2`	%i = i (complexe)
`--> abs(1+%i)`	module
`--> x=5+3`	stocke le résultat dans la variable x
`--> x^2`	
`--> y=3+4;`	point-virgule : effectue le calcul sans afficher le résultat
`--> y`	
`--> y=5;y^2`	plusieurs calculs sur la même ligne
`--> y=1+2+3+4+5+6+...`	un calcul sur plusieurs lignes
`--> 7+8+9+10`	
`--> v=[1,2,3]`	vecteur ligne
`--> w=[1;2;3]`	vecteur colonne
`--> v'`	transposition
`--> u=[4,5,6];[v,u]`	concaténation de deux vecteurs
`--> t=3:7`	début:fin
`--> t=1:2:10`	début:pas:fin
`--> t=0:0.1:1`	
`--> linspace(0,10,4)`	4 nombres régulièrement espacés entre 0 et 10
`--> u=t+1`	
`--> u=2*t`	
`--> u+t`	
`--> u*t'`	produit scalaire
`--> A=[1,2;3,4;5,6]`	matrice
`--> B=[1,2,3;...`	
`--> 4,5,6;...`	
`--> 7,8,9]`	
`--> size(A)`	=[nb de lignes, nb de colonnes]
`--> ones(3,4)`	
`--> ones(A)`	matrice de même dimension que A
`--> zeros(3,4)`	
`--> eye(5,5)`	matrice identité (I prononcé à l'anglaise)
`--> rand(3,4)`	matrice aléatoire (uniforme entre 0 et 1)

`--> diag([1 2 3])`	matrice diagonale
`--> diag(B)`	diagonale de B
`--> M=[ones(2,2),zeros(2,3);...`	construction d'une matrice par blocs
`--> 2*ones(3,2),eye(3,3)]`	
`--> 2*A+1`	
`--> A+A`	
`--> A*[1,2,3;4,5,6]`	multiplication matricielle
`--> A+B`	dimensions non compatibles !
`--> B(3,2)`	élément à la ligne 3, colonne 2
`--> B(:,2)`	deuxième colonne
`--> B(2,:)`	deuxième ligne
`--> B(2:3,1:2)`	matrice extraite : intersection des lignes 2 à 3 avec les colonnes 1 à 2
`--> B(3,2)=10 ;`	change la valeur de l'élément ligne 3 colonne 2
`--> B`	
`--> B(:,2)=[-1;-3;-5];`	change la deuxième colonne
`--> B`	

Exercice 1
Construire les matrices suivantes :

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 2 & 1 & -1 & -1 & -1 \\ 2 & 2 & -1 & -1 & -1 \\ 2 & 3 & -1 & -1 & -1 \end{pmatrix}$$

$$D = \begin{pmatrix} 2 & 2 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$E = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

`--> u=2*%pi*rand()`	nombre aléatoire dans [0,2π]
`--> w=[cos(u),sin(u)]`	
`--> norm(w)`	norme
`--> norm(w,1)`	norme L^1
`--> v=sin([0:%pi/2:2*%pi])`	opération vectorielle
`--> [m,k]=maxi(v)`	maximum m=v(k)
`--> [m,k]=mini(v)`	
`--> sign(v)`	signe
`--> 1==0`	F = False (== égal)
`--> 1~=0`	T = True (~= différent)
`--> 1==0 & 1~=0`	et
`--> 1==0 | 1~=0`	ou
`--> v>0`	vecteur booléen (teste chaque élément de v)

`--> v>=0`	
`--> bool2s(v>=0)`	T = 1, F = 0
`--> v(v>=0)`	extrait les éléments positifs ou nuls
`--> w=1:9; sum(w)`	somme
`--> cumsum(w)`	somme cumulée
`--> A=rand(2,3);sin(A)`	applique sin sur chaque élément de A
`--> sum(A,'c')`	somme sur les lignes (résultat = colonne)
`--> sum(A,'r')`	somme sur les colonnes (résultat = ligne = « row »)
`--> A'`	transposition
`--> rank(A)`	rang
`--> A'*A`	multiplication matricielle
`--> eye(A)`	matrice identité des dimensions de A
`--> ans*A`	
`--> A=[1,2;3,4];det(A)`	déterminant
`--> exp(A)`	exponentielle de chaque élément
`--> expm(A)`	exponentielle matricielle
`--> w=1:2:9`	
`--> w($)`	dernier élément
`--> w($-1)`	avant-dernier élément
`--> int(5.2)`	partie entière

Exercice 2

- taper `help inv`
- calculer l'inverse d'une matrice carrée A de dimension 5 dont les éléments sont des entiers pris au hasard entre 0 et 5 (inclus)
- vérifier avec A*inv(A) et norm(A*inv(A)-eye(A))

Opérations terme à terme

`--> x=[1,2,3];x.*x`	multiplication vectorielle terme à terme
`--> y=[-6,12,8];x.*y`	
`--> A=rand(2,3)`	
`--> B=ones(A);A.*B`	multiplication matricielle terme à terme
`--> y=1 ./x`	division terme à terme ; attention à l'espace car 1./x = 1.0/x
`--> x.*y`	
`--> A=rand(2,3)`	
`--> B=rand(A)`	
`--> A./B`	

Exercice 3

- Créer une matrice aléatoire (uniforme dans [0,1]) à 5 lignes et 3 colonnes. Multiplier la première colonne par 5, la deuxième par 2 et la troisième par 7 en utilisant `ones` et la multiplication terme à terme.
- Faire la même opération en utilisant `diag`.

Valeurs propres

`--> A=eye(5,5);spec(A)`	spectre (= ensemble des valeurs propres)
`--> spec(1,2;3,4)`	
`--> A=[1,2;-3,4];`	
`--> [Ab,X]=bdiag(A)`	bloc-diagonalisation Ab = matrice diagonalisée X = matrice de passage

Exercice 4

- Créer une matrice aléatoire à valeurs dans [0,1] de dimension (4,4). Diagonaliser dans R puis dans C (taper `help bdiag`).

Graphiques

```
--> x=0.1:0.1:10;y=sin(x)./x;
--> plot2d(x,y)                          graphique
--> z=cos(x)./x;
--> clf();                               efface le graphique (clear figure)
--> plot2d(x,y);plot2d(x,z)              affiche 2 graphiques superposés
--> clf();plot2d([x;x]',[y;z]')          idem
--> clf();plot2d(x,z,style=-5)
--> clf();                               rect=[xmin,ymin,xmax,ymax]
plot2d(x,z,rect=[0,0,4,4])
```

Scripts

Un script est un fichier contenant une liste de commandes Scilab. Pour créer un nouveau script, cliquez sur « Editor » dans le menu. Tapez le script suivant dans l'éditeur:

```
t=linspace(0,1,10);
x=sin(t.^2);
plot2d(t,x);
```

Enregistrez sous le nom « `script1.sce` » (menu File > Save as). Revenez dans Scilab et exécutez le script ainsi :

```
--> exec("script1.sce")
```

Exercice 5

- modifier le script pour afficher 20 points au lieu de 10

Programmation

Ecrivez les programmes suivants dans des scripts et exécutez-les.

Conditions

```
x=rand()*1.1;
if x<0.5 then p="pile";                  si … alors
elseif x<1 then p="face";                sinon si … alors
else p="tranche";                        sinon
end                                      fin du si
p
```

Boucle while

```
x=rand();
n=0;
while x<0.9 do                           tant que … faire
n=n+1;
x=rand();
end                                      fin de la boucle
```

Boucle for

```
n=1:10;
m=1;
for i=n                                  pour i prenant chaque valeur du vecteur n
m=m*i;
end                                      fin de la boucle
```

Fonctions

Les fonctions sont définies dans des scripts. Voici par exemple la fonction factorielle :

```
function f=factorielle(n)          résultat = nom_de_la_fonction(arguments)
f=prod(1:n);
endfunction                        fin de la fonction
```

Enregistrez le fichier sous le nom `factorielle.sci` (les fichiers de scripts ont généralement l'extension .sce, les fichiers de fonctions ont l'extension .sci), puis exécutez-le (`exec('factorielle.sci')`). Vous pouvez maintenant utiliser la nouvelle fonction factorielle :

```
--> factorielle(4)
--> factorielle(8)
```

Si l'on modifie la fonction dans le fichier, il faut enregistrer et recharger le fichier dans Scilab (`exec`) avant de pouvoir utiliser la fonction (sinon on utilise l'ancienne version).

Une fonction peut renvoyer plusieurs résultats et prendre plusieurs arguments (scalaires, vectoriels ou matriciels):

```
function [x,y]=polaire(theta,r)
x=r*cos(theta);
y=r*sin(theta);
endfunction
```

Une fonction peut s'appeler elle-même (elle est dite *récursive*) :

```
function f=factorielle2(n)
if n<=1 then
  f=1;
else
  f=n*factorielle2(n-1)
end
endfunction
```

Exercice 6

- écrire la fonction de Heavyside qui renvoie 1 si l'argument est positif ou nul et 0 sinon
- écrire une fonction qui recherche le plus petit élément d'un vecteur en utilisant `for`
- écrire une fonction récursive qui recherche le plus petit élément d'un vecteur
- écrire une fonction qui donne le nième terme de la suite de Syracuse définie par :
  u_{n+1} = $u_n/2$ si u_n est pair
  = $3u_n + 1$ sinon
  (connaissant le terme initial u_0)
- d'après la conjecture de Collatz, quel que soit le terme initial, la suite de Syracuse finit toujours par boucler sur le cycle 4, 2, 1, 4, 2, 1… Ecrire une fonction qui pour un terme initial u_0 donné, renvoie n tel u_n=1

Pour en savoir plus

- le site web du cours : http://www.di.ens.fr/~brette/calculscientifique/index.htm
- le site web de Scilab : http://www.scilab.org/
- documentation : http://www.scilab.org/publications/index_publications.php?page=books.html