Initiation à Scilab

1 Introduction

Scilab est l'abréviation de "*Scientific laboratory*". Il s'agit d'un environnement informatique conçu spécifiquement pour le calcul matriciel. De même que son "grand frère" MATLAB (malheureusement, un produit commercial !), dont il reprend les principes, ce logiciel est parfaitement adapté à l'analyse de circuits, au traitement du signal et des images, à la conception de filtres, à l'analyse de systèmes de commande, et à bien d'autres domaines d'application. De plus, sa facilité d'emploi avec les nombres complexes et les tracés graphiques en fait un outil intéressant dans de nombreux problèmes de programmation. Scilab peut d'ailleurs être vu comme un langage de programmation de quatrième génération. Il peut être utilisé dans un mode interactif où les instructions sont exécutées immédiatement après qu'elles aient été tapées. Inversement, un programme peut être écrit à l'avance et sauvegardé sur le disque dans un fichier, à l'aide d'un éditeur, puis exécuté sous Scilab. Les deux modes sont utiles.

Signalons que le logiciel peut être téléchargé gratuitement à l'adresse `http://www.scilab.org`.

2 Vecteurs et matrices

Scilab est spécialement conçu pour manipuler des matrices. La manière la plus simple d'entrer une matrice est d'utiliser une ligne explicite d'éléments. Dans la liste, les éléments sont séparés par des blancs ou des virgules, et des point virgules (;) sont utilisés pour indiquer la fin de ligne. La liste est encadrée par des crochets `[ ]`. Par exemple, l'instruction

```
>> A = [1 2 3;4 5 6;7 8 9]
```

fournit comme résultat

```
A =
   1  2  3
   4  5  6
   7  8  9
```

La variable `A` est donc une matrice de dimension 3×3. Les éléments d'une matrice peuvent être formés de n'importe quelle expression Scilab. Par exemple, l'instruction

```
>> x = [-1.3 sqrt(3) (1+2+3)*4/5]
```

fournit

```
x =
   -1.3000 1.7321 4.8000
```

Une matrice avec une seule ligne ou une seule colonne est un vecteur, et une matrice 1×1 est un scalaire. Les éléments d'une matrice peuvent être référencés par leurs indices placés entre parenthèses. Tapez `x(5)=abs(x(1))` pour créer le nouveau vecteur

```
x =
   -1.3000 1.7321 4.8000 0 1.3000
```

Notez que la dimension de `x` a été automatiquement ajustée pour tenir compte du nouvel élément, et que les éléments non référencés sont pris par défaut égaux à zéro (ici `x(4)`). De nouvelles lignes ou de nouvelles colonnes peuvent être ajoutées très facilement. Essayez de taper `r = [10 11 12], A = [A;r]`. Les dimensions doivent coïncider dans l'instruction. Essayez `r = [13 14], A = [A;r]`.

La commande `size(A)` fournit le nombre de lignes et le nombre de colonnes de `A`. `size(A)` est elle-même une matrice de taille 1×2. Ces nombres peuvent être mémorisés si nécessaire par la commande `[m n] = size(A)` Dans l'exemple précédent, `A = [A;r]` est une matrice 4×3, la variable `m` contient donc le nombre 4 et `n` le nombre 3. Un vecteur est une matrice pour laquelle soit `m`, soit `n` est égal à 1. Si `m` est égal à 1, la matrice est un vecteur ligne, si `n` est égal à 1, la matrice est un vecteur colonne. Les matrices et les vecteurs peuvent avoir des éléments complexes. Par exemple, la commande

```
>> A = [1 2;3 4]+%i*[5 6;7 8]
```

et la commande

```
>> A = [1+%i*5 2+%i*6;3+%i*7 4+%i*8]
```

sont équivalentes, elles produisent toutes deux la matrice

```
>> A =
      1.0000+5.0000i 2.0000+6.0000i
      3.0000+7.0000i 4.0000+8.0000i
```

Scilab contient de nombreuses expressions prédéfinies pour la manipulation de matrices. Le caractère spécial (`'`) sert à désigner la transposée d'un matrice. La commande `A = [1 2 3;4 5 6;7 8 9]'` produit la matrice

```
A =
   1 4 7
   2 5 8
   3 6 9
```

Les lignes de `A'` sont les colonnes de `A`, et vice versa. Si `A` est une matrice complexe, `A'` est sa transposée conjuguée, ou transposée hermitienne. Pour obtenir une transposée non conjuguée, il faut employer les deux caractères point-prime (`.'`). Les variables matricielles et vectorielles peuvent être ajoutées, soustraites, et multipliées comme des variables ordinaires à condition que leur dimension soit correcte. Seules les matrices de même dimension peuvent être ajoutées ou soustraites. Il existe cependant une manière simple de soustraire le même scalaire à tous les éléments d'une matrice. Par exemple, `x = [1 2 3 4], x = x-1` produit le résultat

```
x =
   1 2 3 4

x =
   0 1 2 3
```

La multiplication de deux matrices n'a de sens que si leurs dimensions "internes" sont égales. En d'autres termes, `A*B` n'est valable que si le nombre de colonnes de `A` est égal au nombre de lignes de `B`. Si a_{ij} désigne l'élément situé sur la $i$ème ligne et la $j$ème colonne, alors la matrice `A*B` est formée des éléments

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{1}$$

où n est le nombre de colonnes de `A` et aussi le nombre de lignes de `B`.

Essayez de taper `A = [1 2 3;4 5 6]; B = [7;8;9]; A*B`. Vous devriez obtenir le résultat suivant

```
ans =
     50
    122
```

Le produit interne (produit scalaire) de deux vecteurs colonnes `x` et `y` est le scalaire défini comme le produit `x'*y` ou, de manière équivalente `y'*x`. Par exemple, `x = [1;2],y = [3;4], x'*y` conduit au résultat

```
ans =
    11
```

La norme hermitienne d'un vecteur est définie comme la racine carrée du produit interne du vecteur avec lui même. On peut aussi la calculer à l'aide de la fonction `norm`. Essayez par exemple de calculer la norme du vecteur `[1 2 3 4]`. Vous devriez obtenir `5.4772`.
Le produit externe, ou antiscalaire, de deux vecteurs colonnes est la matrice antiscalaire `x*y'`. De même, le produit externe de deux vecteurs lignes est la matrice `x'*y`.
Tout scalaire peut être multiplié par une matrice. La multiplication se fait alors élément par élément. Ainsi, `A = [1 2 3;4 5 6;7 8 9];A*2` donne le résultat suivant

```
ans =
    2  4  6
    8 10 12
   14 16 18
```

Vérifier que `2*A` donne le même résultat.

L'inverse d'une matrice est calculée à l'aide de la fonction `inv(A)` qui n'est valide que si `A` est carrée. Si la matrice est singulière, ou non inversible, un message d'erreur est affiché. Si vous tapez `inv(A)` avec la matrice précédente, vous devriez obtenir

```
 1.0E+15 *

   1.576106   - 3.1522119    1.576106
 - 3.1522119    6.3044238  - 3.1522119
   1.576106   - 3.1522119    1.576106
```

Une matrice peut ne pas être singulière, mais son nombre de conditionnement `rcond(A)` être si petit que la précision du calcul de l'inverse est sujette à caution. Pour vous en convaincre, calculez les valeurs propres de `A`, en tapant `spec(A)`. L'inverse d'une matrice est utilisée pour résoudre un système d'équations linéaires. Ainsi, pour résoudre le système

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & -2 & 4 \\ 0 & -2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} \tag{2}$$

vous pouvez taper `A = [1 2 3;1 -2 4;0 -2 1]; b = [2;7;3]; inv(A)*b` pour obtenir

```
ans =
    1
   -1
    1
```

vérifiez que la réponse est correcte en tapant `A*ans`. Qu'observe t'on ?

Scilab offre un autre moyen de résoudre un système linéaire, basé sur l'élimination de Gauss, qui est plus rapide que le calcul d'une l'inverse. La syntaxe est `A\b`. Elle est valable tant que `A` possède le même nombre de lignes que `b`. Essayez.

L'opérateur "point". En calcul matriciel, on a souvent besoin d'effectuer une opération élément par élément. Dans Scilab, ces opérations sont appelées opérations sur des réseaux ou des tableaux (*array operations*). Bien entendu, l'addition et la soustraction sont des opérations qui se font élément par élément. L'opération `A.*B` désigne la multiplication, élément par élément, des matrices `A` et `B`. Construisez arbitrairement deux matrices 3×3, `A = rand(3,3)` et `B = rand(3,3)`, et tapez successivement

```
>> A*B
>> A.*B
```

Qu'observe t'on ? Supposons que nous voulions calculer le carré de chaque élément de `A`. La bonne manière de spécifier ce calcul est

```
>> A_square = A.^2
```

où le point indique qu'une opération doit être effectuée sur chaque élément de `A`. Sans ce point, `A` est multipliée par `A` suivant les règles usuelles de la multiplication des matrices, ce qui conduit à un résultat totalement différent :

```
>> A^2
```

Essayez également

```
>> 2.^A
```

et comparez avec

```
>> 2^A
```

3 Le deux points

On peut utiliser le deux points de différentes manières dans Scilab (voir `help colon`). Il sert fondamentalement à construire un vecteur dont les valeurs des éléments sont incrémentées séquentiellement. Tapez par exemple

```
>> x = 3:9
```

pour obtenir

```
x =
   3 4 5 6 7 8 9
```

L'incrément est de 1 par défaut. Pour avoir un autre incrément, tapez des commandes telles que

```
>> x = 1:0.5:4
>> x = 6:-1:0
```

La plupart des fonctions Scilab acceptent des entrées vectorielles et produisent des sorties vectorielles. La commande

```
>> y = sqrt(1:10)
```

construit un vecteur d'entiers de 1 à 10 et prend la racine carrée de chacun d'entre eux. Essayez-la.

Autre subtilité : quel est l'effet de chacune de ces deux commandes et pourquoi ?

```
>> 1+1:5
>> 1+(1:5)
```

Ce dernier exemple montre qu'il faut s'efforcer de taper les commandes de manière non ambigüe.

Augmentation d'une matrice ou d'un vecteur. Les dimensions d'une matrice ou d'un vecteur peuvent être augmentées en introduisant de nouveaux éléments empruntés à une autre matrice ou un autre vecteur. Soit par exemple le vecteur `x = [1 3 5]`. La commande

```
>> x = [x 6 8 10]
```

fournit le résultat suivant

```
x =
   1  3  5  6  8  10
```

De même la commande

```
>> y = [x;1:6]
```

donne

```
y =
   1  3  5  6  8  10
   1  2  3  4  5   6
```

Extraction d'une sous-matrice. On peut aussi utiliser les deux points pour extraire une sous-matrice d'une matrice `A`.

`A(:,j)`	extrait la $j$ème colonne de `A`. On considère successivement toutes les lignes de `A` et on choisit le $j$ème élément de chaque ligne.
`A(i,:)`	extrait la $i$ème ligne de `A`.
`A(:)`	reforme le matrice `A` en un seul vecteur colonne en concaténant toutes les colonnes de `A`.
`A(j:k)`	extrait les éléments j à k de `A` (balayée colonne par colonne) et les stocke dans un vecteur.
`A(:,j:k)`	extrait la sous-matrice de `A` formée des colonnes j à k.
`A(j:k,:)`	extrait la sous-matrice de `A` formée des lignes j à k.
`A(j:k,q:r)`	extrait la sous-matrice de `A` formée des éléments situés dans les lignes j à k et dans les colonnes q à r.

Ces définitions peuvent s'étendre à des pas d'incrémentation des lignes et des colonnes différents de 1.

Conversion de type. Notons enfin que Scilab considére par défaut les variables qui sont créées comme des flottants en double précision. Il peut cependant être parfois intéressant de les transformer en entiers 8 bits non signés. Vérifiez le résultat de l'opération de conversion

```
>> T = rand(16,8)*255;
>> whos -name T
>> T=uint8(T);
>> whos -name T
```

4 Variables complexes

Le nombre complexe $\sqrt{-1}$ est prédéfini dans Scilab et stocké dans la variable `%i`. Cette variable commence par le caractère `%` pour indiquer qu'elle est prédéfinie et que son contenu ne peut être modifié. Examiner aussi le contenu de la variable `%pi`.

Notez la manière dont est affichée une variable complexe. Entrez par exemple `z1 =1+2*%i`, `z2 = 2+1.5*%i`. Comme `%i` est considéré comme une variable, il faut utiliser le signe `*` de la multiplication, sinon apparaîtra un message d'erreur. Scilab ne fait pas de différence entre les variables réelles et une variable complexe (sinon à la mise en mémoire). Les variables peuvent être ajoutées, soustraites, multipliées et même divisées. Tapez par exemple `x = 2, z = 4 + 5*%i` et `z/x`. Les parties réelle et imaginaire de `z` sont toutes deux divisées par `x`. Scilab traite simplement `x` comme une variable dont la partie imaginaire est nulle. Une variable complexe dont la partie imaginaire est nulle est traitée comme une variable réelle. Soustrayez `2*%i` de `z1` et affichez le résultat.

Scilab contient plusieurs fonctions prédéfinies pour manipuler les nombres complexes. Par exemple, `real(z)` extrait la partie réelle du nombre complexe `z`. Tapez

```
>> z = 2+1.5*%i; real(z)
```

pour obtenir le résultat

```
ans =
    2
```

De même, `imag(z)` extrait la partie imaginaire du nombre complexe `z`. La fonction `abs(z)` calcule le module du nombre complexe `z`. Tapez par exemple

```
>> z = 2+2*%i;
>> r = abs(z)
>> theta = imag(log(z)) // calcul de l'argument du nombre complexe
>> z = r*exp(%i*theta)
```

La dernière commande montre comment retrouver le nombre complexe original à partir de son module et de sa phase.
Une autre fonction utile, `conj(z)`, retourne le complexe conjugué du nombre complexe `z`. Si `z = x+%i*y` où `x` et `y` sont réels, alors `conj(z)` est égal à `x-%i*y`. Vérifiez ceci pour différents nombres complexes en utilisant la fonction `conj(z)`.

5 Notions plus avancées

Scilab permet de gérer des tableaux de plus de 2 dimensions. Ceux-ci peuvent être définis de manière directe. Par exemple, pour obtenir un tableau de dimensions $2 \times 4 \times 3$, on peut procéder de la manière suivante :

```
>> T = hypermat([2 4 3]);
>> T(:,:,1) = [1 2 3 4;5 6 7 8];
>> T(:,:,2) = [9 10 11 12;13 14 15 16];
>> T(:,:,3) = [17 18 19 20;21 22 23 24]
```

Pour vérifier le nombre de dimensions de ce tableau, et préciser le nombre d'éléments suivant chaque dimension, faites appel à la commande `size(T)`.

Certaines fonctions permettent de générer automatiquement des tableaux multidimensionnels. Essayez la commande `T=ones(4,5,23)`. Observez le type de données générées à l'aide de la commande `whos()`.

Scilab offre la possibilité de manipuler des structures de données plus complexes que les tableaux de flottants. Par exemple, on peut créer des listes, chaque élément de la liste pouvant correspondre à un tableau de nature et de dimensions différentes. Pour illustrer cette fonctionnalité, tapez les lignes de commandes suivantes :

```
>> C = list('toto');
>> C(2) = %pi;
>> C(3) = [1 2 3];
>> C(4) = zeros(3,4,5);
```

On accède alors aux éléments de la liste de façon similaire à ceux d'un vecteur usuel. Ainsi, vous pouvez examiner le contenu de `C(4)`. Essayer aussi de supprimer un élément de la liste :

```
>> C(3) = null();
```

6 Aide

Scilab possède une aide (help) en ligne et toute une collection de démonstrations. Pour obtenir des détails sur les aides disponibles, tapez

```
>> help
```

Pour une aide sur une fonction spécifique, `sin` par exemple, tapez

```
>> help sin
```

Pour avoir des informations sur le '`;`', tapez

```
>> help semi
```

Pour apprendre à tracer des courbes en Scilab, tapez

```
>> help plot2d
```

Comment peut-on visualiser l'allure de la fonction tangente hyperbolique sur l'intervalle $[-5, 5]$? Mettre un titre au graphique avec la commande `xtitle`.

Pour rechercher les fonctions relatives à un mot clef donné (en anglais), on peut utiliser la commande `apropos`. Par exemple, essayez

```
>> apropos('root')
```

Pour avoir une vision globale des fonctionnalités de Scilab, vous pouvez lancer quelques démos.