TD N°1 – Introduction au langage Matlab

Gilles Simon and Stéfane Paris

{Gilles.Simon,Stefane.Paris}@loria.fr

Pourquoi Matlab ?

Matlab fournit un ensemble de fonctions opérant sur des tableaux multidimensionnels ; le son numérique peut être décrit par un vecteur dont les valeurs sont les échantillons temporels le décrivant ; l'image numérique peut également être représentée par une matrice d'échantillons. Ainsi, le son et l'image sont facilement manipulables à l'aide des fonctions et des outils fournis par Matlab. Les matrices et vecteurs sont codés sous forme de tableaux à une ou deux dimensions respectivement. Il suffit alors d'apprendre à manipuler les tableaux sous Matlab pour effectuer des prototypages rapides de traitements dédiés au son ou à l'image. Matlab possède également l'avantage d'être un interprète et non un compilateur. On peut donc tester nombre d'instructions avant même de construire la moindre fonction ou le moindre script.

Le noyau

Les noms des variables respectent les règles usuelles de construction. Il faut noter que Matlab distingue les minuscules des majuscules. Le point d'accueil en début de ligne de la fenêtre de commande (le *prompt* par anglicisme) est de la forme `>>`.

Vous retrouverez les fonctions classiques :

`floor(`x`)` retourne la partie entière de x,

`ceil(`x`)` retourne la partie entière +1 de x,

`round(`x`)` retourne l'arondie de x,

`log(`x`)` retourne le logarithme naturel de x.

Attention 1 *Une fonction classique, telle que* `round`*, appliquée à une matrice revient à l'appliquer à chacun des éléments de cette matrice.*

```
>> A = [[ 0.5, 0.7, 0.5, 0.7];[0.4, 0.8, 0.4, 0.8]]
>> round(A)
```

Par la suite, on sera souvent amené à vouloir appliquer le même traitement à un sous-ensemble des éléments d'une matrice. Par exemple, si l'on veut doubler (*2) la valeur des éléments de la matrice `A` qui sont en première ligne et en première et deuxième colonnes, on opère comme suit :

```
>A(1,1:2) = 2 * A(1,1:2)
```

Exercice 1 *Créez la matrice* 5×6 *suivante :*

$$X = \begin{pmatrix} 2 & 5 & 6 & 8 & 7 & 3 \\ 7 & 8 & 7 & 9 & 3 & 1 \\ 1 & 4 & 6 & 2 & 0 & 0 \\ 6 & 6 & 5 & 3 & 2 & 4 \\ 0 & 8 & 0 & 6 & 5 & 9 \end{pmatrix}$$

Puis, en une seule et unique instruction, extrayez la sous-matrice centrale de A, constituée de toutes les lignes et de toute sles colonnes de A sauf les premières et dernières :

$$\text{ssX} = \begin{pmatrix} 8 & 7 & 9 & 3 \\ 4 & 2 & 6 & 0 \\ 6 & 5 & 3 & 2 \end{pmatrix}$$

Multipliez cette matrice par trois (3) et réinsérez la dans X :

$$X = \begin{pmatrix} 2 & 5 & 6 & 8 & 7 & 3 \\ 7 & 24 & 21 & 27 & 9 & 1 \\ 1 & 12 & 18 & 6 & 0 & 0 \\ 6 & 18 & 15 & 9 & 6 & 4 \\ 0 & 8 & 0 & 6 & 5 & 9 \end{pmatrix}$$

Les génératrices de matrices

Exercice 2 *Testez les fonctions suivantes :*

`zeros(M,N)` *génère une matrice nulle de taille $M \times N$ dont les éléments ont pour valeur 0,*

`ones(M,N)` *génère une matrice unaire de taille $M \times N$ dont les éléments ont pour valeur 1,*

`rand(M,N)` *génère une matrice aléatoire de taille $M \times N$ dont les éléments ont des valeurs aléatoires dans l'intervalle [0, 1] de distribution uniforme,*

`eye(M,N)` *génère une matrice identité de taille $M \times N$.*

Les attributs

`size(A)` vecteur contenant le nombre de lignes et de colonnes de la *matrice A*,

`size(A,1)` nombre de lignes de la *matrice A*,

`size(A,2)` nombre de colonnes de la *matrice A*,

`length(A)` indique le nombre d'éléments de l'objet *A*,

`max(A)` retourne le maximum de l'objet *A*,

`min(A)` retourne le minimum de l'objet *A*,

Exercice 3 *Suivant une démarche similaire à celle de l'exercice 1, définissez la matrice suivante :*

$$X = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{pmatrix}$$

Multipliez par quatre (4) la sous-matrice des éléments centraux de taille moitié de la matrice X. Le résultat doit être proche de ceci :

$$X = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 12 & 12 & 12 & 12 & 12 & 12 & 3 & 3 & 3 \\ 4 & 4 & 4 & 16 & 16 & 16 & 16 & 16 & 16 & 4 & 4 & 4 \\ 5 & 5 & 5 & 20 & 20 & 20 & 20 & 20 & 20 & 5 & 5 & 5 \\ 6 & 6 & 6 & 24 & 24 & 24 & 24 & 24 & 24 & 6 & 6 & 6 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \end{pmatrix}$$

Transposition

La transposition d'une matrice est assurée par l'opérateur post-fixé `'`.

Par exemple `X'` vaut :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 12 & 16 & 20 & 24 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

La concaténation et la réduction

Dans le même esprit, vous pouvez contruire une matrice à partir d'une autre.

Par exemple : `W = X([1:M/4,3*M/4+1:M],[1:N/4,3*N/4+1:N])` où `M` et `N` sont les nombres de lignes et de colonnes de `X` respectivement.

Exercice 4 *Construisez une matrice binaire de taille* 10×10 *ayant la forme d'un damier où les valeurs* ***-1*** *et* ***+1*** *alternes successivement. Vous ne devez utiliser que la génératrice* ($\text{ones}(M, N)$) *de matrices unitaires et la génération* ($\text{eye}(M, N)$) *de matrices identités, avec* $\{M, N\} \leqslant 5$. *Le résultat doit ressembler à ceci :*

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{pmatrix}$$

Besoin d'aide ? vous devez utiliser un des opérateurs présentés dans la documentation matlab (`>> doc`) à l'endroit suivant : MATLAB → Mathematics → Linear Algebra → Matrix Operations

La transformée de Fourier rapide

La fonction ci-dessous constuit un signal bruité échantilloné à `sample_rate` Hz contenant deux fréquences pures `F1` et `F2`, puis calcule la transformée de Fourier discrète de ce signal :

```
function test_fft(sample_rate, F1, F2)
% Composantes fréquentielles d'un signal
% ----------------------------------
% Construction d'un signal bruité échantilloné à sample_rate hz
%  contenant deux fréquences pures F1 et F2 Hz.
figure;

t = [0:1/sample_rate:0.6];
N = length(t); % nombre d'échantillons
s = sin(2*pi*F1*t) + sin(2*pi*F2*t+pi/4);
subplot(2,2,1); plot(t,s);

y = fft(s);
f = sample_rate*(0:(N-1))/N; % vecteur de fréquences associé
n = length(f);
subplot(2,2,2); plot(f,abs(y(1:n)));

sb = s + normrnd(0,0.1,1,N); % on ajoute un peu de bruit (n'hésitez pas à tester avec + de 0.1)
subplot(2,2,3); plot(t,sb);

y = fft(sb);
f = sample_rate*(0:(N-1))/N;
n = length(f);
subplot(2,2,4); plot(f,abs(y(1:n)));
end
```

On ne vous demande pas de comprendre tous les rouages de la FFT à ce stade du cours, mais de tester la fonction avec différentes valeurs de `sample_rate`, `F1` et `F2`, afin d'avoir une première approche, intuitive, de cette transformation. Essayez par exemple :

```
>> test_fft(1000,50,70)
>> test_fft(1000,50,700)
>> test_fft(2000,50,700)
```

Ensuite, modifiez le code pour qu'il affiche le spectre uniqumemnt sur la moitié de la plage fréquentielle (entre 0 et $F_{\max}/2$) *Qu'observez-vous ?*