TP 1 - Introduction à Matlab

Pour une documentation plus conséquente, vous pouvez taper `doc` dans une fenêtre Matlab.

I- Quelques manipulations basiques sous Matlab

Tapez `Matlab` dans une fenêtre de commandes.

1-Calculatrice

Voici quelques exemples de calculs que vous pouvez taper :

```
>> 2*3+4e5
>> x=ans; y=sqrt(x)
>> z=1+i
>> z^2, w=pi+j
```

Remarquez que la virgule et le point virgule permettent de séparer des commandes sur une ligne et que le point virgule supprime l'affichage du résultat.

Notez aussi `pi`, ... pour noter le nombre π, et `i, j`.

Voici quelques fonctions usuelles (essayez-les):

`abs, exp, log, cos, sin, tan, acos, asin, atan, cosh, sinh, tanh, acosh, asinh, atanh, sqrt, floor, ceil.`

2-Aide

Pour obtenir la description d'une commande Matlab, vous pouvez taper `help commande` ou utiliser la fenêtre d'aide. Par exemple,

```
>> help format.
```

Essayez ensuite

```
>> format long, pi
>> format long e, pi
```

puis revenez au format par défaut.

3-Calcul matriciel

Matlab permet de faire des calculs matriciels très rapides. Il faut utiliser cette "force" lors de l'écriture des programmes que vous voulez lui faire exécuter. Notez que, pour Matlab, les variables réelles et complexes sont des matrices (tableaux) 1×1.

```
>> x=2; x(1), x(1,1)
```

Définition de vecteurs :

`>> b=[1+i,3,5]` (vecteur ligne)

`>> c=[1-i;3;5]` (vecteur colonne)

`>> b', b.'` (adjoint, transposé)

Définition de matrices :

```
>> a=[1,2,3;4,5,6]
>> abis=[1,2,3;...
4,5,6]
```

`>> a=[a;7,8,10]` (concaténation)

`>> d=[c,a,c,c;b,b]`

`>> a*b, a*c, b*a, c'*a` (produit)

`>> d(1,1), d(1,2), d(2,1)` (éléments)

Quelques matrices particulières :

```
>> rand(1,3), rand(size(a))
```

(Consultez l'aide sur la commande `rand`).

```
>> zeros(1,3), ones(size(c)), eye(5,5), eye(size(a))
>> diag(a), diag(c), diag(a,1), diag(c,1)
```

Remarquez les différentes utilisations possibles de `diag`.

Opérations matricielles :

```
>> a^2, inv(a), rank(a)
>> a(3,3)=9
>> eig(a), [v,d]=eig(a)
```

Créations de suites arithmétiques :

```
>> -3:3, y=2:-.3:-2.4, y(4), y(2:5)
>> A=[1:6;2:7;4:9;20:25], A(:,1), A(1,:)
>> A(3:4,:), A(1:2,2: 2:6)
>> linspace(-1,1,10)
>> [nl,nc]=size(a)
>> length(a)
```

Autres opérations sur les matrices :

```
>> a=[1:3;4:6;7:9]
```

`>> a.^2, a.*a, 1 ./a, 1./a` (opérations élément par élément)

Les fonctions usuelles sont appliquées élément par élément :

```
>> exp(a), sin(a)
```

Les fonctions de matrices sont

```
>> expm(a), logm(a), sqrtm(a)
```

`>> sum(a,1), sum(a,2), prod(a,1), prod(a,2)` (sommation, produit par ligne ou colonne)

```
>> norm(a)
```

4-Représentations graphiques

Voici un exemple (allez voir dans l'aide pour l'utilisation de `plot2d`):

```
>> x=linspace(-pi,2*pi,100)'; y=sin(x); plot(x,y)
```

Voir aussi les fonctions `plot3, surf, ezsurf, mesh, griddata, meshgrid`. Essayez ce qui suit:

```
>> x=rand(100,1)*4-2; y=rand(100,1)*4-2; z=sin(x.*y);
>> ti=-2:0.25:2; [XI,YI]=meshgrid(ti,ti); ZI=griddata(x,y,z,XI,YI);
>> mesh(XI,YI,ZI), hold on; plot3(x,y,z,'o'), hold off;
```

5-Programmation

On dispose des boucles habituelles : `for, while, if, switch`. Allez voir dans l'aide pour la syntaxe.

Un programme est défini dans un fichier script, saisi avec un éditeur de texte, qui contient la suite d'instructions que vous voulez faire exécuter par Matlab. On donne à ce fichier le suffixe `.m` (pour se souvenir qu'il s'agit d'un script Matlab), par exemple `script1.m`. Pour exécuter le fichier, tapez dans une fenêtre Matlab le nom du fichier sans le suffixe

```
>> script1
```

6-Définition de fonctions

Un fichier de fonction contient une seule fonction. Le fichier doit avoir le même nom que la fonction auquel on ajoute le suffixe `.m`. Le texte du fichier doit commencer par le mot `function`. La définition d'une fonction commence comme suit

```
function [y1,y2]=f(x1,x2,x3)
```

où `x1,x2,x3` sont les variables d'entrée, `y1,y2` les variables de sortie et `f` le nom de la fonction. On peut définir plusieurs fonctions dans le même fichier, mais elle ne seront visibles qu'à l'intérieur de ce fichier. (Une fonction est chargée en mémoire au premier appel, si elle est modifiée et que vous n'obtenez pas la nouvelle version, taper `clear` pour effacer la mémoire). Pour passer une fonction en paramètre d'une autre fonction on utilise un pointeur de fonction : `@nom_de_la_fonction` (cela remplace l'ancienne méthode: `nom_de_la_fonction` et `feval`).

II- Exemples

1-Calcul matriciel

a) Écrire une fonction qui définit la matrice de taille $n \times n$

$$A = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}$$

Quelle est l'utilité de l'instruction `sparse`?

b) Soit `X` un vecteur ou une matrice défini(e) dans Matlab. Ecrivez l'instruction qui permet de calculer le vecteur ou la matrice `Y`, de même taille que `X`, dont l'élément (i, j) est égal à $f(X_{i,j})$, dans chacun des cas suivants :

(1) $f(x) = (x-1)(x+4)$,
(2) $f(x) = |2x^2 - 3x + 1|$,
(3) $f(x) = 1/(1+x^2)$,

c) Soient deux vecteurs de même taille, par exemple `X=Y=linspace(-1,1,50)`. En utilisant la commande `meshgrid`, construisez la matrice `A` telle que $A_{i,j} = f(x_j, y_i)$, dans le cas où $f(x, y) = x \exp(y)$. Tracer la surface $z = f(x, y)$ et les courbes de niveau de f (commandes `surf`, `surfc`, `contour`, `contour3`).

2-Résolution approchée d'une EDP

Créez deux fichiers `u0.m` et `a0.m` contenant la définition des deux fonctions suivantes :

```
function [y]=u0(x)
% donnee initiale
y=zeros(size(x));
for j=1:length(x)
   if (x(j)<=-1) y(j)=0;
   elseif (x(j)<=-0.75) y(j)=4*(x(j)+1);
   elseif (x(j)<=-0.5) y(j)=-4*(x(j)+1/2);
   else y(j)=0;
   end
end
function [y]=a0(x)
% coefficient
y=ones(size(x));
```

Visualisez `u0` en tapant

```
>> x=linspace(-1,1,100)'; plot(x,u0(x));
```

Créer un troisième fichier contenant le script suivant. Exécutez-le en comprenant ce qui se passe.

```
%Resolution approchee de l'equation de transport du/dt+a(x)du/dx=0
%avec donnee initiale u(0,x)=u0(x)
clear
T=1.5;X=1;
dt=0.02; dx=0.025; l=dt/dx;
N=floor(T/dt); J=floor(2*X/dx);
t=linspace(0,T,N+1); x=linspace(-X,X,J+1);
```

```
u=zeros(N+1,J+1);
a=a0(x);
u(1,:)=u0(x);
j=2:J;
for n=1:N
   u(n+1,j)=u(n,j)-l*a(j) .*(u(n,j)-u(n,j-1));
[XI,TI]=meshgrid(x,t);
mesh(XI,TI,u);xlabel('Axe x');ylabel('Axe t');view(-15,30);
end
```

Avec le script précédent, vous visualisez $u(t,x)$ comme une surface dans $\mathbb{R}^3$. On peut aussi faire un graphe animé:

```
% Représentation de u(t,x) sous forme animée:
% Construction de l'animation
for n=1:N+1
   clf;plot(x,u(n,:));ylim([0 1]);F(n)=getframe;
end
close;
% Lancement de l'animation
movie(F);close;
```