Modélisation et résolutions numérique et symbolique de problèmes via les logiciels Maple et MATLAB (MODEL)

Cours n°1 : Introduction à MATLAB

Stef Graillat

Université Pierre et Marie Curie (Paris 6)

Format du cours

Équipe pédagogique : Stef Graillat

Évaluation des connaissances

- un examen réparti n°1 (25%), un examen réparti n°2 (60%) et une note de TD/TME (15%)
- 10 séances de TME (dont certains seront à rendre)

Horaire

- Cours : le jeudi de 13h30 à 15h30 (salle 42-43/411)
- TD/TME : le vendredi de 13h30 à 15h30 (salle 41/117) et de 15h45 à 17h45 (salle 31/308)

Site web :
`http://www-pequan.lip6.fr/~graillat/teach/model/index.html`

Objectifs du cours

Objectifs :

- Concept mathématique : définition mathématique de concepts et de quantités
- Algorithme : comment calculer efficacement ces quantités sur ordinateur (via l'utilisation de MATLAB et Maple) ?
- Résolution de problème : utiliser les concepts et les algorithmes pour résoudre des problèmes concrets

Références principales

- Scientific Computing with Case Studies, Dianne P. O'Leary, SIAM, 2009
- Numerical Computing with MATLAB, Cleve Moler, SIAM, 2004
- MATLAB Guide, Desmond J. Higham, Nicholas J. Higham, 2nd édition, SIAM, 2005
- Solving Problems in Scientific Computing Using Maple and MATLAB, Walter Gander, Jiri Hrebicek, 4e édition, Springer, 2004
- Numerical Recipes. The Art of Scientific Computing, William Press, Saul Teukolsky, William Vetterling et Brian Flannery, 3rd Edition, Cambridge University Press, 2007

Champs d'application

Les notions vues dans ce cours interviennent dans :

- la robotique
- le traitement du signal
- le traitement d'image
- la géométrie algorithmique
- la biologie
- etc.

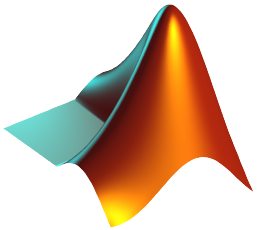
Calcul formel / calcul numérique

- Algorithmes symboliques : fournissent une représentation exacte du résultat mais peuvent être lents
- Algorithmes numériques : exécutés en précision finie, ils sont plus rapides en général mais fournissent des résultats approchés

→ calcul symbolique-numérique : utiliser quand cela est possible des algorithmes numériques (au sein d'algorithmes symboliques) en contrôlant la précision des résultats

Logiciels utilisés

Calcul numérique
- MATLAB
- SciLab
- Octave

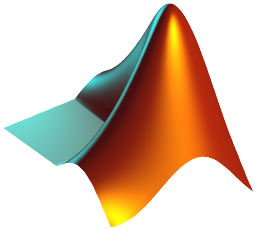


Calcul formel
- Maple
- Sage
- Maxima
- Mathematica
- Magma

Plan général du cours

1. Présentation et introduction à MATLAB
2. Décomposition en valeurs singulières, application à la compression d'image
3. Calcul de vecteurs propres, valeurs propres, application à l'algorithme PageRank de Google
4. Transformée de Fourier discrète, application en traitement du signal et en calcul formel
5. Méthode de Monte-Carlo, application au pricing d'option en finance
6. Introduction à Maple
7. Optimisation, somme de carrés, méthode de Newton, application à la stabilité des systèmes dynamiques
8. Optimisation formelle : méthode des extrémas liés
9. Problème de classification, application à la robotique
10. Comptage et isolation de racines de polynômes, application au tracé de courbes certifiées

1. Introduction à MATLAB



MATLAB

- MATLAB = MATrix LABoratory
- un langage de programmation et un environnement de développement
- MATLAB a été conçu par Cleve Moler à la fin des années 1970
- MATLAB est complété par de multiples boites à outils (toolboxes)
- le langage MATLAB supporte la POO
- Intéraction possible avec les langages C et Fortran

- pour l'aide sur une commande `command`, utiliser `help command` ou `doc command`
- pour écrire des commentaire %

Référence : MATLAB Guide, Desmond J. Higham, Nicholas J. Higham, 2nd édition, SIAM, 2005

Vecteurs et matrices en MATLAB

Les variables de MATLAB sont principalement des vecteurs et des matrices

- Les vecteurs :
  - vecteur ligne

```
>> format compact
>> x = [1.1 10.1 100.1]
x =
   1.1000 10.1000 100.1000
```

  - vecteur colonne

```
>> x = [1.1; 10.1; 100.1]
x =
   1.1000
  10.1000
 100.1000
```


Vecteurs en MATLAB (suite)

- Affichage et affectation

```
>> x = [1.1; 10.1; 100.1];
>> x
x =
   1.1000
  10.1000
 100.1000
>> x(3) = -1.1
x =
   1.1000
  10.1000
  -1.1000
```

Les vecteurs sont indicés à partir de 1 (et pas 0 comme en C)

Vecteurs en MATLAB (suite)

- Transposition d'un vecteur

```
>> x = [1.1 10.1 100.1]'
x =
   1.1000
  10.1000
 100.1000
```

- Pour entrer un vecteur ou une commande occupant plus d'une ligne

```
>> x = [0 .05 .10 .15 .20 .25 .30 .35 .40 .45 .50 ...
   .55 .60 .65 .70 .75 .80 .85 .90 .95 1];
```

- Longueur d'un vecteur

```
>> length(x)
ans =
   21
```


Vecteurs en MATLAB (suite)

- Vecteur de taille quelconque (par défaut les vecteurs sont en ligne)

```
n = 20;
h = 1/n;
for k=1:n
   x(k) = k*h;
end
```

- Initialisation d'un vecteur colonne

```
x = zeros(n,1);
```

- les deux-points. La notation `a:b` dénote le vecteur ligne allant de `a` à `b` par pas de 1 tandis que pour `a:s:b` le pas est `s`

```
>> x = 1:5
x =
   1 2 3 4 5
>> x = 4:-1:0
x =
   4 3 2 1 0
```


Vecteurs en MATLAB (suite)

- `>> x = 0:0.05:1;  % vecteur à 21 composantes.`
  `>> x = 0.05*(0:20) % autre façon de générer le même vecteur.`
- Accéder à des parties d'un vecteur
  `x(1:4)` extrait les 4 premiers éléments de `x`
- `linspace(a,b,n)` produit un vecteur ligne avec n composantes qui divise [a,b] en $n-1$ intervalles égaux.

```
x = linspace(0,1,21);
```


Les matrices en MATLAB

La puissance de MATLAB provient de ses opérations matricielles

- rapide
- et précise (qualité numérique)

2 façons d'entrer des matrices

```
>> A = [1 2 3
        2 4 7
       -1 0 5]
A =
   1 2 3
   2 4 7
  -1 0 5
>> a = [1 2 3; 1 4 8; 3 -1 0]
a =
   1 2 3
   1 4 8
   3 -1 0
```

Les matrices en MATLAB (suite)

Fonctions spéciales pour créer des matrices

- `zeros(m,n)` produit une matrice de 0 de taille $m \times n$

```
>> A = zeros(2,4)
A =
   0 0 0 0
   0 0 0 0
```

- `ones(m,n)` produit une matrice de 1 de taille $m \times n$

```
>> A = ones(3)
A =
   1 1 1
   1 1 1
   1 1 1
```


Les matrices en MATLAB (suite)

- `eye(n)` produit la matrice identité de taille n

```
>> A = eye(3)
A =
   1 0 0
   0 1 0
   0 0 1
```

- Résolution d'un système linéaire $Ax = b$

```
>> A = [1 2 3; 2 4 7; -1 0 5];
>> b = [1 1 1]';
>> x =A\b
x =
    -6
     5
    -1
```


Les matrices en MATLAB (suite)

- Autre utilisation des deux-points :

```
>> C = A([1 3],:)
C =
     1     2     3
    -1     0     5
```

- Fonctions « vectorisées »
  On veut évaluer une fonctions sur un vecteur de valeur x_i :
  $a = x_1 < x_2 < \cdots < x_n = b$.
  Les fonctions standard acceptent des vecteurs en argument et revoie un vecteur.

```
n=21;
x = linspace(0,2*pi,n);
y = cos(x);
```


Sauvegarder votre travail

- En général, on écrit la liste des commandes tapées dans un fichier texte (via l'éditeur `edit` intégré à MATLAB) ou via votre éditeur de texte préféré

- Pour sauvegarder des variables, utilisez la fonction `save`. Sauvegarder les variable `A`, `b` dans le fichier `svar.mat` se fait par :

```
save svar A b
```

On les recharge par la commande :

```
load svar
```

Aide

- taper `help command`.

```
>> help length
 LENGTH   Length of vector.
    LENGTH(X) returns the length of vector X.  It is
    equivalent to MAX(SIZE(X)) for non-empty arrays
    and 0 for empty ones.
```

- pour l'avoir en mode graphique, taper `doc`

```
>> doc length
```


M-files

2 types de fichier (qui portent l'extension `.m`) :

- script M-files : ni entrée, ni sortie et utilise les variables de l'espace de travail
- fonction M-files : contient une fonction qui accepte des arguments en entrée et renvoie des arguments en sortie et les variables internes sont locales à la fonction

Exemple de fonction (à stocker dans un fichier `sumprod.m`) :

```
function [s,p]= sumprod(x)
   n = length(x);
   s=0;
   p=1;
   for i=1:n
      s = s + x(i);
      p = p*x(i);
   end
```


M-files (suite)

Structure d'une fonction M-files

1. le mot-clé `function`
2. liste des arguments en sortie (entre crochets `[ ]` s'il y en a plusieurs)
3. le symbole =
4. le nom de la fonction (qui doit être le même que le nom du fichier `.m`)
5. le liste entre parenthèse des entrées
6. le corps de la fonction

Pour éditer les fichiers, taper `edit`

Commandes utiles : `dir`, `ls`, `cd`, `type`, `lookfor`, `path`

Affichage

- Format numérique : commande `format` pour afficher des nombres en virgule fixe ou flottante

```
>> format short, pi^4 % fixe, 5 chiffres
ans =
   97.4091
>> format short e, pi^4 % flottante, 5 chiffres
ans =
   9.7409e+001
>> format long, pi^4 % fixe, 15 chiffres
ans =
   97.40909103400242
>> format long e, pi^4 % flottante, 15 chiffres
ans =
   9.740909103400242e+001
```

Affichage (suite)

- Affichage de chaînes de caractère
  - la commande `disp` permet d'afficher une chaîne de caractère ou une variable

```
>> x=3;
>> disp(x);
     3
>> disp('test');
test
```

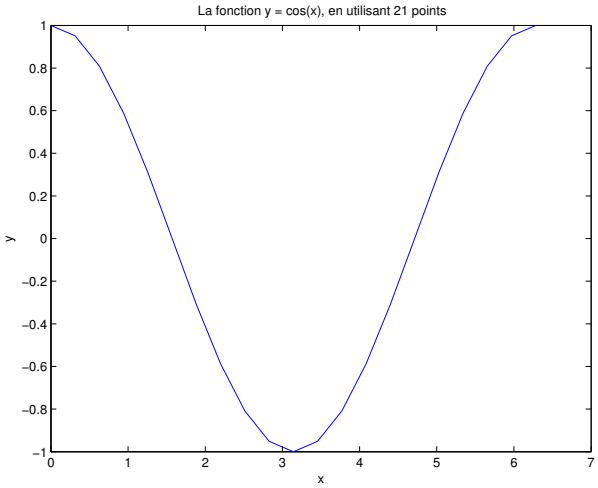
  - la commande `fprintf` permet d'afficher les chaînes de caractère et des chiffres (similaire à la commande `printf` de C)

Graphiques

Tracé d'une fonction $y = f(x)$ sur un intervalle $[a, b]$

```
n = 21;
x = linspace(0,2*pi,n);
y = cos(x);
plot(x,y)
title('La fonction y = cos(x), en utilisant 21 points')
xlabel('x')
ylabel('y')
```


Graphiques



Structures de contrôle et tests

- boucle `for`

```
for variable = expression
   instructions
end
```

- boucle `while`

```
while expression
   instructions
end
```

- test `if`

```
if expression
   instructions
end
```

```
if expression
   instructions
else
   instructions
end
```

Structures de contrôle et tests

- opérations relationnelles

==	égal
~=	différent
<	strictement inférieur
>	strictement supérieur
<=	inférieur ou égal
>=	supérieur ou égal

- opérations logiques

&	et
|	ou
~	non

Symbolic Math Toolbox

- La toolbox contient le noyau de Maple et des fonctions MATLAB qui communiquent avec ce noyau
- Nouveau type d'objet : les objets `sym` créés par les commandes `sym` et `syms`
- Créer une variable symbolique `x`

```
>> syms x
```


Symbolic Math Toolbox (suite)

- Manipulation symbolique d'expression en `x`

```
>> f = 1/(1+x^2)
f =
   1/(1+x^2)

>> g = int(f) % intégration
g =
   atan(x)

>> diff(g) % dérivation
ans =
   1/(1+x^2)

>> syms a
>> y = solve(f-a) % résoud f(x)-a=0
y = [ 1/a*(-a*(-1+a))^(1/2)]
    [ -1/a*(-a*(-1+a))^(1/2)]
```


Symbolic Math Toolbox (suite)

- Arithmétique multiprécision : fonction `vpa`

```
>> digits % par défaut
Digits = 32
>> vpa('sqrt(2)')
ans =
1.4142135623730950488016887242097
>> digits(50)
>> vpa('sqrt(2)')
ans =
1.4142135623730950488016887242096980785696718753769
```

- Arithmétique exacte : on travaille avec des expressions symboliques

```
>> z = sym('sqrt(2)')
z =
sqrt(2)
>> z^2-2
ans =
0
```

MATLAB et Maple à l'ARI

- Il y a 16 licences MATLAB à l'ARI. Il s'agit de MATLAB R2009a (version 7.8) avec la Symbolic Math Toolbox.

  Pour lancer MATLAB, il suffit de taper `matlab` dans un terminal

- Il y a Maple à l'ARI. Il s'agit de Maple 10.

  Pour lancer Maple, il suffit de taper `/usr/local/maple10/bin/xmaple`

Conclusion

Pour avoir un aperçu des fonctionnalités de MATLAB, taper `demo`