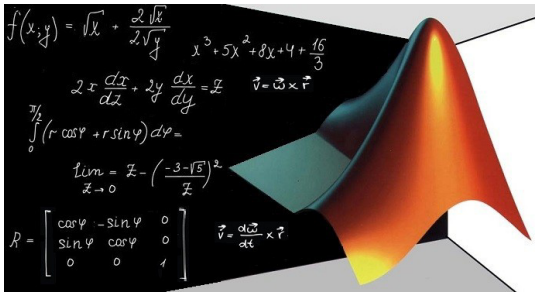
MATLAB pour débutants



Djedoui Nassim

Février 2019
1.0

Table des matières

Objectifs

A l'issu de ces deux chapitres, l'étudiant sera capable de :

- Acquérir les notions de base de programmation (pour les débutants)
- D'apprendre le plat de forme du logiciel MATLAB.
- Découvrir une vision cohérente du langage MATLAB (Pour les anciens utilisateurs de MATLAB).
- De programmer quelques fonctions de bases tels que le factoriel, la somme, la puissance moyenne et efficace d'un signal et la résolution d'un système d'équation linéaire.

Pré-requis : L'étudiant doit avoir:

- Pour bien suivre ce cours, l'étudiant doit avoir des connaissances sur les langages classiques tels que le langage Pascal. De même, il faut avoir un petit bagage sur les algorithmes et la programmation.
- Pouvoir traduire les problème réel en un langage de programmation.

Introduction

MATLAB[p.35 AA] est un système interactif et convivial de calcul numérique et de visualisation graphique destiné aux ingénieurs et scientifiques. Il possède un langage de programmation à la fois puissant et simple d'utilisation. Il permet d'exprimer les problèmes et solutions d'une façon aisée, contrairement aux autres langages de programmation.

MATLAB intègre des fonctions d'*analyse numérique*, de *calcul matriciel*, de *traitement de signal*, de *visualisation graphique 2D et 3D*, etc. Il peut être utilisé de façon interactive ou en mode programmation. En mode interactif, l'utilisateur a la possibilité de réaliser rapidement des calculs sophistiqués et d'en présenter les résultats sous forme numérique ou graphique. En mode programmation, il est possible d'écrire des scripts (programmes) comme avec d'autres langages. L'utilisateur peut aussi créer ses propres fonctions pouvant être appelées de façon interactive ou par les scripts. Ces fonctions fournissent à MATLAB un atout inégalable : son extensibilité. Ainsi, l'environnement MATLAB peut être facilement étendu.

Dans MATLAB, l'élément de base est la matrice. L'utilisateur ne s'occupe pas des allocations mémoire ou de redimensionnement comme dans les langages classiques. Les problèmes numériques peuvent être résolus en un temps record, qui ne représente qu'une fraction infime du temps à passer avec d'autres langages comme le Basic, C, C++ ou le Fortran.

MATLAB s'impose dans les mondes universitaire et industriel comme un outil puissant de simulation et de visualisation de problèmes numériques. Dans le monde universitaire MATLAB est utilisé pour l'enseignement de l'algèbre linéaire, le traitement du signal, l'automatique, ainsi que dans la recherche scientifique. Dans le domaine industriel, il est utilisé pour la résolution et la simulation de problèmes pratiques d'ingénierie et de prototypage. MATLAB est une abréviation de MATrix LABoratory. Ecrit à l'origine, en fortran, par Cleve Moler, MATLAB était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle, écrite en C par The MathWorks Inc., existe en version "professionnelle" et en version "étudiant". Sa disponibilité est assurée sur plusieurs plates-formes : Sun, Bull, HP, IBM, compatibles PC, Macintosh, et plusieurs machines parallèles. MATLAB est conforté par une multitude de boîtes à outils (Toolboxes) spécifiques à des domaines variés. Un autre atout de MATLAB, est sa portabilité ; la même portion de code peut être utilisée sur différentes plates-formes sans la moindre modification.

MATLAB possède les particularités suivantes par rapport au d'autres langages :

- la programmation facile,
- la continuité parmi les valeurs entières, réelles et complexes,
- la gamme étendue des nombres et leur précision,
- la bibliothèque mathématique très compréhensive,
- l'outil graphique qui inclut les fonctions d'interface graphique et les utilitaires,
- la possibilité de liaison avec les autres langages classiques de programmation (C ou Fortran).

Le premier chapitre est dédié pour la prise en main de MATLAB alors que le deuxième chapitre traite la notion de programmation sous fichier ainsi que les fonctions.

Exercice

[solution n°1 p.33]

C'est quoi MATLAB ?

- ☐ Un Langage de programmation
- ☐ Un système d'exploitation
- ☐ Un logiciel interactif de calcul matriciel

Chapitre 1

1. Objectifs du chapitre 1

Dans ce Chapitre nous allons parcourir l'interface de MATLAB jusqu'au outils graphiques passant par les définitions de base.

2. L'environnement MATLAB

MATLAB affiche au démarrage plusieurs fenêtres. Selon la version on peut trouver les fenêtres suivantes :

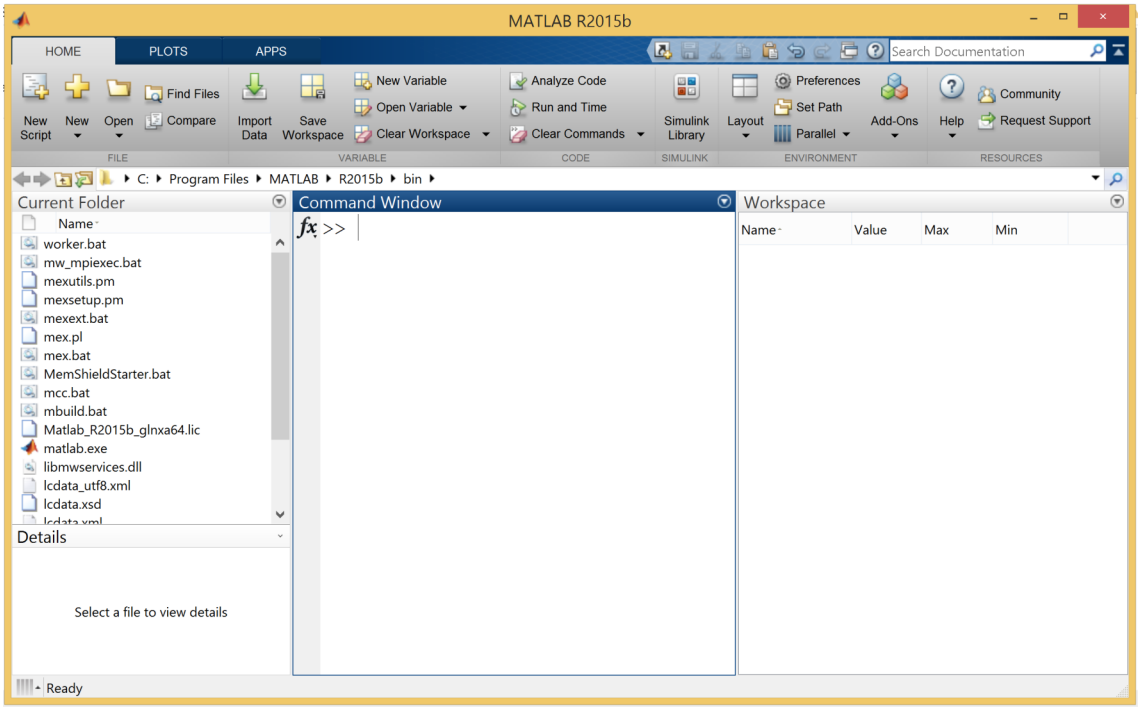


Figure 1 : L'environnement MATLAB

Noter les définitions suivantes :

- Current Folder: indique le répertoire courant ainsi que les fichiers existants.
- Workspace: indique toutes les variables existantes avec leurs types et valeurs.
- Command History: garde la trace de toutes les commandes entrées par l'utilisateur.
- Command Window: utilisée pour formuler nos expressions et interagir avec MATLAB. C'est la fenêtre que nous utilisons tout au long de ce chapitre.

3. Types de données

Dans MATLAB, il y a un seul type de données : le type matrice (Matrix). Tout est matrice, un scalaire est une matrice carrée d'ordre 1. Il n'y a donc pas de déclaration de types. De même, l'utilisateur ne s'occupe pas de l'allocation mémoire. Les variables matrices et vecteurs peuvent être redimensionnés et même changer de type.

⚠ *Attention*

MATLAB fait la distinction entre les majuscules et minuscules ; ainsi, mavar et MAVAR sont deux variables différentes. Utiliser l'exemple suivant pour voir la différence...

```
>> variable=2;
>> Variable
??? Undefined function or variable 'Variable'>
```

4. Notions de base de MATLAB

4.1. Le mode interactif

Dans le mode interactif, MATLAB peut être utilisé comme une "super-puissante" calculatrice scientifique. On dispose des opérations arithmétiques et d'un ensemble important de fonctions de calcul numérique et de visualisation graphique.

4.2. Variables spéciales et constantes

Dans MATLAB, on trouve des constantes pré-définies :

pi : 3.14159265358979.

eps : 2.2204e-016 (distance entre 1.0 et le flottant le plus proche) ; cette valeur peut être modifiée,

Inf (Infinite) : nombre infini,

NaN (Not a Number) : n'est pas un nombre, exprime parfois une indétermination.

ans : variable contenant la dernière réponse.

Comme constantes prédéfinies, on trouve :

```
>> pi
ans = 3.1416

>> 1/0
Warning: Divide by zero
ans =
 Inf

>> 0/0
Warning: Divide by zero
ans =
 NaN
```

4.3. Opérations arithmétiques

Les opérations arithmétiques de base dans MATLAB sont : + pour l'addition, - pour la soustraction, * pour la multiplication et / ou \ pour la division. La division à droite et la division à gauche ne donnent pas les mêmes résultats, ce sont donc deux opérations différentes que ce soit pour les scalaires ou pour les vecteurs et matrices.

Exemple : Exécuter les commandes suivantes :

```
>> 5+7-3+2*5
ans =
 19
a/b : c'est a qui est divisé par b
>> 4/3
ans =
 1.3333
```

```
a\b : c'est b qui est divisé par a
>> 4\3
ans =
 0.7500
```

Les opérations peuvent être enchaînées en respectant les priorités usuelles des opérations et en utilisant des parenthèses.

```
>> 4*(-5)+12
ans =
 -8
>> 2.3*(4-6)/(3+15)
ans =
 -0.2556
```

MATLAB possède l'opérateur "^" pour l'élévation à une puissance entière ou réelle.

```
>> 3^4
ans =
 81
```

4.4. Format des nombres et précision des calculs

MATLAB a la possibilité d'afficher les valeurs des variables dans différents formats :

flottants courts (*short*), flottants longs (*long*), flottants courts en notation scientifique *(short e*), flottants longs en notation scientifique (long e), ainsi que les formats monétaire, hexadécimal et rationnel (notation sous forme d'une fraction).

Format flottant short

```
>> format short
>> 22/7
ans =
 3.1429
```

Format flottant long

```
>> format long
>> 22/7
ans =
 3.142857142857143
```

format flottant short en notation scientifique

```
>> format short e
>> 22/7
ans =
 3.1429e+000
```

format flottant long en notation scientifique

```
>> format long e
>> 22/7
ans =
 3.142857142857143e+000
```

format rationnel (fraction)

```
>> format rat
>> 0.36
ans =
 9/25
>> format hex
>> 1977
ans =
 409ee40000000000
```

4.5. Edition des lignes de commande

MATLAB conserve l'historique des commandes entrées de façon interactive lors d'une session. Il est donc possible de récupérer des instructions déjà saisies et de les modifier dans le but de les réutiliser.

Il suffit de double-cliquer dessus pour les réexécuter. On peut tout aussi les effacer et les supprimer de l'espace de travail par un clic droit de la souris. L'historique est conservé dans la fenêtre History de l'environnement MATLAB.

5. Tableaux

5.1. Vecteurs ou tableaux à 1 dimension

Le moyen le plus simple de saisir un vecteur de type ligne est d'entrer ses éléments en les séparant par des blancs ou des virgules. Définir le vecteur x qui contiens les valeurs 6 4 7,,

Saisie du tableau x, vecteur ligne

```
>> x = [6 4 7]
x =
 6 4 7
```

Le programme précédent a permit de créer le vecteur x qui est de la forme du tableau suivant :

x	6	4	7

Tableau 1 : Valeurs d'un vecteur sous forme d'un tableau

Afin d'éviter l'affichage du résultat d'une expression quelconque, on terminera celle-ci par un point-virgule.

Autre façon de saisir un vecteur ligne

```
>> x = [6,4,7] ;
```

Ce vecteur est considéré comme une matrice à une ligne et trois colonnes.

```
>> size(x)
ans =
 1 3
```

Les dimensions d'un tableau quelconque peuvent être récupérées par la commande size sous forme

d'un vecteur, [m n], m et n étant respectivement le nombre de lignes et de colonnes.

```
>> [m n] = size(x)
m =
 1
n =
 3
```

La longueur d'un tableau quelconque est, par définition, sa plus grande dimension.

```
>> longueur_x = length(x)
longueur_x =
 3
```

5.2. Construction d'un tableau à partir d'un autre

```
>> v = [x 1 2 8]
v =
 6 4 7 1 2 8
```

De même pour avoir le vecteur w = [1 6 4 7 1 2 8], nous avons deux possibilités :

```
>> w = [1 v]
w =
 1 6 4 7 1 2 8
>> w = [1 x 1 2 8]
w =
 1 6 4 7 1 2 8
```

Dans MATLAB, les indices d'un tableau commencent à 1. Pour récupérer une composante

d'un vecteur, il faut spécifier son indice entre parenthèses.

```
>> w(3)
ans =
 4
```

Si les composantes d'un vecteur sont espacées d'un pas constant et si la première et la dernière valeur sont connues, alors ce vecteur peut être décrit de la manière suivante :

```
>> debut = 0; fin = 1;
>> pas = 0.25;
>> t = debut:pas:fin
t =
 0 0.2500 0.5000 0.7500 1.0000
```

Si on omet de spécifier ce pas, celui-ci est pris par défaut à 1.

```
>> x=3:6
x =
 3 4 5 6
```

L'addition et la soustraction de vecteurs de mêmes dimensions se font élément par élément.

```
>> x = [0 4 3];
>> y = [2 5 7];
>> x-y
```

```
ans =
 -2 -1 -4
>> x+y
ans =
 2 9 10
```

Ajouter ou retrancher un scalaire à un vecteur, revient à l'ajouter ou le retrancher à toutes les composantes du vecteur.

```
>> 3+x
ans =
 3 7 6
>> x-2
ans =
 -2 2 1
```

De même, la multiplication et la division d'un vecteur par un scalaire sont réalisées sur toutes les composantes du vecteur.

```
>> 2*x
ans =
 0 8 6
>> x/4
ans =
 0 1.0000 0.7500
```

La transformation d'un vecteur ligne en un vecteur colonne et inversement, sera réalisée à l'aide de l'opérateur de transposition'.

```
>> tx = x'

tx =
 0
 4
 3

```

Le produit d'un vecteur par sa transposée donne le carré de la norme de celui-ci.

```
>> x*tx
ans =
 25
```

Le produit d'un vecteur colonne de taille n par un vecteur ligne de taille m donne une matrice de dimensions (n, m).

```
>> tx*y
ans =
 0 0 0
 8 20 28
 6 15 21

```

En précédant d'un point les opérateurs *, / , \ et ^, on réalise des opérations élément par élément.

```
>> x.*y
ans =
 0 20 21
```

```
>> x.^2
ans =
 0 16 9
>> x./y
ans =
 0 0.8000 0.4286
>> y.\x
ans =
 0 0.8000 0.4286
```

Plusieurs fonctions opérant directement sur des vecteurs sont disponibles sous MATLAB.

On peut en citer :

- `sum` : somme des composantes d'un vecteur,
- `prod` : produit des composantes d'un vecteur,
- `sqrt` : racines carrées des composantes d'un vecteur,
- *mean* : moyenne des composantes d'un vecteur.

```
>> sum(x)
ans =
 7

>> prod(y)
ans =
 70

>> sqrt(x)
ans =
 0 2.0000 1.7321

>> mean(x)
ans =
 2.333
>> cumsum(x) % somme cumulée des composantes du vecteur x
ans =
 0 4 7

```

5.3. Matrices ou tableaux à 2 dimensions

La matrice ou tableau à 2 dimensions est un type de base de MATLAB.

La saisie d'une matrice peut être faite de différentes façons.

- Vecteurs lignes séparés par un saut de ligne (;)

```
>> matrice=[5 7 9; 1 4 2]
matrice =
 5 7 9
 1 4 2
```

- Vecteurs séparés par un retour à la ligne

```
>> matrice2=[3 5 7
8 6 10]
matrice2 =
```

```
4  3 5 7
5  8 6 10
```

Soit les matrices A et B suivantes :

```
1 >> A =[5 7 3;2 9 4]
2 A =
3  5 7 3
4  2 9 4
5 >> B = [5 3 0; 7 1 6]
6 B =
7  5 3 0
8  7 1 6
```

La concaténation horizontale se fait comme suit :

```
1 >> ConcatHoriz = [A B]
2 ConcatHoriz =
3  5 7 3 5 3 0
4  2 9 4 7 1 6
```

La concaténation verticale se fait en faisant suivre A par un retour à la ligne (pointvirgule).

```
1 >> ConcatVert = [A; B]
2 ConcatVert =
3  5 7 3
4  2 9 4
5  5 3 0
6  7 1 6
```

6. Les chaînes de caractères

Toute chaîne de caractères est une matrice à une ligne, et un nombre de colonnes égal à sa longueur. Une chaîne est considérée par MATLAB comme un vecteur ligne dont le nombre d'éléments est le nombre de ses caractères.

```
1 >> ch = 'matlab'
2 ch =
3 matlab
```

Les dimensions de la chaîne de caractères sont données par la commande size.

```
1 >> [n, m] = size(ch)
2 n =
3  1
4 m =
5  6
6
```

Ainsi la chaîne 'matlab' est considérée comme une matrice à une ligne et six colonnes, ou plus simplement comme un vecteur ligne de taille 6.

La commande `length` qui permet d'obtenir la taille d'un vecteur ou la plus grande dimension d'une matrice, est applicable aux chaînes de caractères dont elle retourne la longueur.

```
1 >> length(ch)
```

```
ans =
 6
```

On peut concaténer plusieurs chaînes de caractères en les écrivant comme des éléments d'un vecteur.

```
>> cch = ['langage ' ch]
cch =
```

7. Les nombres complexes

L'imaginaire pur i ($i^2 = -1$) est noté i ou j. Un nombre complexe est donc de la forme z = a + ib ou a + jb. Mais MATLAB, dans ses réponses, donne toujours le symbole i.

```
>> i^2
ans =
 -1.0000 + 0.0000i
>> j^2
ans =
 -1.0000 + 0.0000i
```

Le conjugué d'un nombre complexe est obtenu par la fonction conj. Considérons le nombre complexe z1.

```
>> z1 = 4-3i
z1 =
 4.0000 - 3.0000i
```

Son conjugué, noté z1c, est :

```
>> z1c = conj(z1)
z1c =
 4.0000 + 3.0000i
```

Le conjugué peut aussi se calculer par la transposition du nombre complexe.

```
>> z1'
ans =
 4.0000 + 3.0000i
```

Nous pouvons aussi effectuer les opérations courantes sur les complexes telles que l'addition, la multiplication, l'élévation à une puissance et la division.

Les fonctions `real` et `imag` permettent d'obtenir respectivement les parties réelle et imaginaire d'un nombre complexe.

```
>> a = real(z1)
a =
 4
>> b = imag(z1)
b =
 -3
```

MATLAB propose les fonctions `abs` et `angle` qui permettent d'obtenir directement le module et l'argument d'un complexe.

```
>> r = abs(z1)
r =
 5
>> theta = angle(z1)
theta =
 -0.6435
```

8. Les polynômes

MATLAB représente un polynôme sous forme d'un tableau de ses coefficients classés dans l'ordre des puissances décroissantes.

8.1. Saisie d'un polynôme

Le polynôme P d'expression suivante, est représenté par le tableau à +1 la dimension su polynôme comme suit

$$p(x) = x^2 - 6x + 9$$

```
>> P = [1 -6 9]
P =
 1 -6 9

```

Le nombre d'éléments du tableau est égal au degré du polynôme +1.

8.2. Racines d'un polynôme

On peut déterminer les racines des polynômes P à l'aide de la fonction `roots`.

```
>> roots(P)
ans =
 3
 3

```

8.3. Détermination d'un polynôme à partir de ses racines

On peut aussi déterminer les coefficients d'un polynôme à partir de ses racines en utilisant la fonction `poly`.

On cherche, par exemple, le polynôme qui a pour racines : 1, 2 et 3.

Celles-ci peuvent être définies comme les éléments d'un vecteur r.

```
>> r = [1 2 3]
```

```
r =
 1  2  3
```

Le polynôme recherché est alors :

```
>> K = poly(r)
K =
 1  -6  11  -6
```

Qui correspond au polynôme K suivant. En multipliant par un réel non nul, tous les coefficients de K, on trouve un autre polynôme ayant les mêmes racines que K.

$$K(x)=x^3-6x^2+11x-6$$

8.4. Multiplication et division de polynômes

La multiplication et la division de polynômes peuvent être réalisées facilement avec MATLAB. Soient deux polynômes P1 et P2 définis par :

$$P1(x)=x+2\,;\,P2(x)=x^2-2x+1$$

```
>> P1 = [1 2]
P1 =
 1  2
```

```
>> P2 = [1 -2 1]
P2 =
 1  -2  1
```

Le résultat de la multiplication de P1 par P2 est le polynôme P3 qui s'obtient avec la fonction `conv`.

```
>> P3 = conv(P1,P2)
P3 =
 1  0  -3  2
```

La division de deux polynômes se fait par la fonction `deconv`. Le quotient `Q` et le reste `R`

de la division peuvent être obtenus sous forme d'éléments d'un tableau.

```

```

```
>> [Q,R] = deconv(P2,P1)
Q =
 1  -4
R =
 0  0  9
```

9. Graphiques 2D et 3D

MATLAB peut produire des graphiques couleurs 2D et 3D impressionnants. Il fournit aussi les outils et moyens de personnaliser et de modifier pratiquement tous leurs aspects, facilement et de manière

parfaitement contrôlée.

Pour avoir un aperçu des possibilités de MATLAB, il est conseillé de consulter le programme de démonstration en entrant la commande `demo` à la suite de l'invite ">>".

```
>> demo matlab graphics
```

9.1. Graphiques 2D

La commande plot permet de tracer des graphiques xy. Avec `plot(x,y)` on trace y en

fonction de x ; x et y sont des vecteurs de données de mêmes dimensions.

Définition de l'intervalle de x et calcul des valeurs de la fonction y = f(x)

```
>> x = -pi:0.1:pi
>> y = sin(x);
```

Tracé de la fonction

```
>> plot(x,y)
```

Documentation du graphique

```
>> grid
>> xlabel('angle : de -\pi à \pi')
>> title('courbe de sin \pi x/\pi x')
```

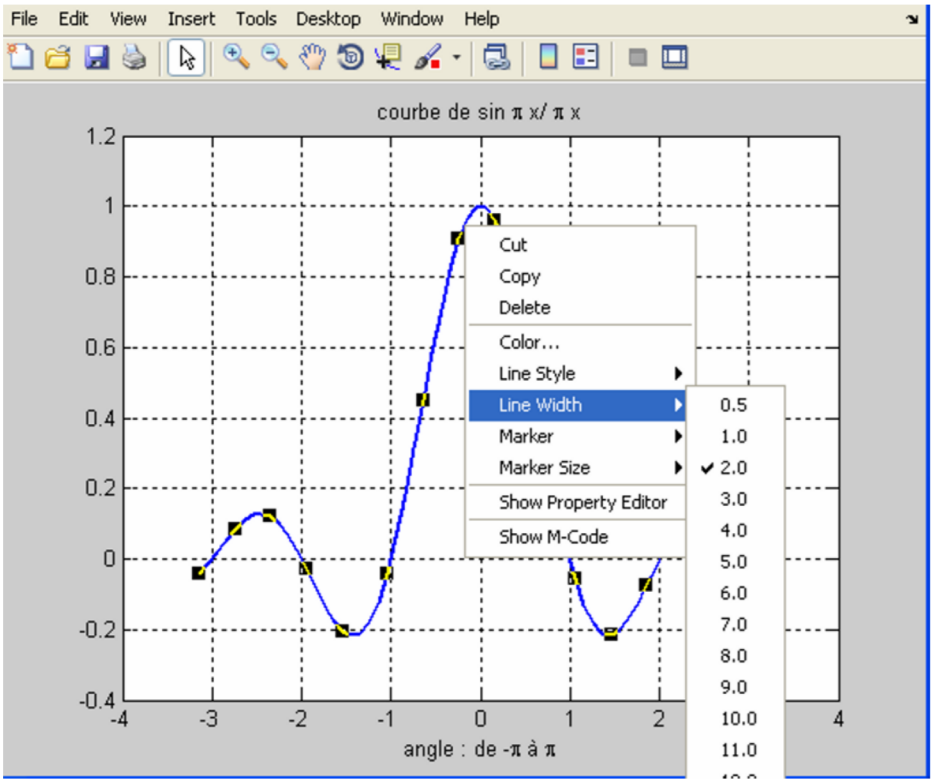


Figure 3 : Exemple de plot en 2D

La taille du trait peut être modifiée graphiquement en le sélectionnant à la souris puis un clic droit en

choisissant l'option `LineWidth` après avoir sélectionné la flèche d'édition graphique.

De même, on peut modifier la taille et la police des caractères du titre et du label de l'axe des x et y ainsi que celles du texte à l'intérieur de la fenêtre graphique.

La commande `figure(gcf)` permet de passer du mode interactif à la fenêtre graphique courante. Vous pouvez aussi utiliser le menu "Window" et sélectionner la fenêtre graphique.

Au graphique courant vous pouvez ajouter un titre, une grille, une légende pour les axes ou du texte sur le graphique grâce aux commandes : `title, grid, xlabel, ylabel, text` et `gtext`.

On peut aussi tracer des courbes paramétriques.

Exemple

```
>> t = 0:0.001:2*pi;
>> x = cos(3*t);
>> y = sin(2*t);
>> plot(x,y)
>> grid
>> xlabel('x')
>> ylabel('y')
>> title('courbe de lissajous')
```

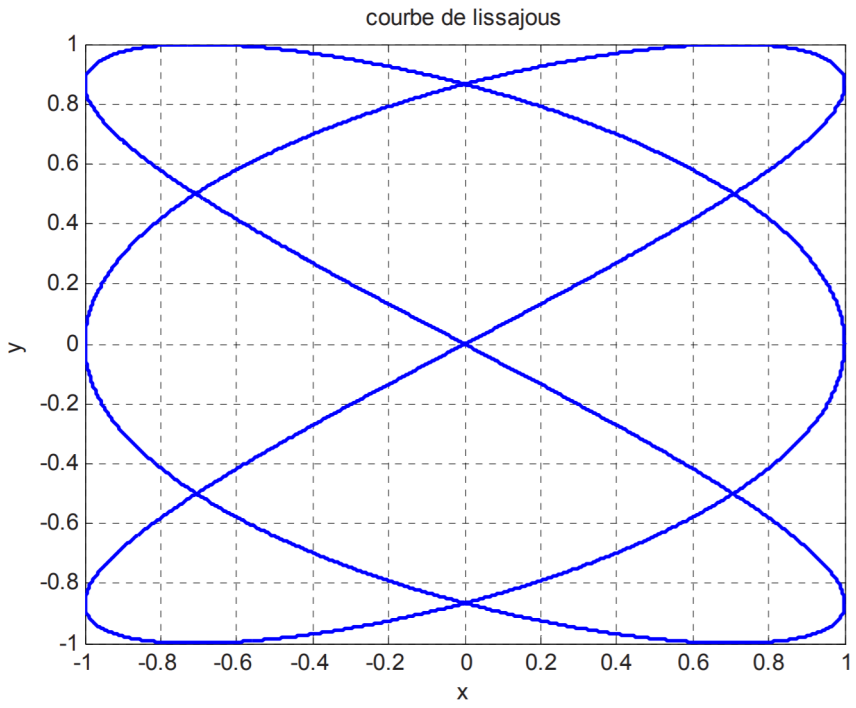


Figure 3 : Exemple 2 du plot en 2D

Pour imprimer le graphique courant, on utilisera la commande `print`. Pour la coller dans un texte, on utilisera l'option `Copy Figure`.

Cette copie se fera selon les options que l'on spécifie dans Copy options de ce même menu *edit*. On peut choisir un arrière-plan blanc (`Force white background`), un arrière-plan transparent, etc. On peut aussi copier l'image comme un *métafile* avec perte ou préservation des informations ou une image

Bitmap (.bmp).

9.2. Graphiques 3D

Soit l'exemple suivant d'une fonction à 2 variables :

$$z = \frac{\sin\left(x^2 + y^2\right)}{x^2 + y^2}$$

pour x et y variant de -π à π avec un pas de π /10.

```
>> alpha = -pi:pi/10:pi;
>> beta = alpha;
```

Dans la prochaine étape on génère deux matrices carrées X et Y qui définissent le domaine de calcul de z, on utilisera pour ceci la fonction `mesh` pour le tracé.

```
[X,Y]= meshgrid(alpha, beta);
```

On évalue la fonction z et on stocke les données dans la variable Z.

```
>> Z = sin(X.^2+Y.^2)./(X.^2+Y.^2);
```

On dessine la surface représentative de la fonction.

```
>> mesh(X,Y,Z)
```

Nous pouvons rajouter un titre pour le tracé (title), des légendes pour les axes (`xlabel, ylabel et zlabel`) ainsi qu'un quadrillage (`grid`).

```
>> xlabel('angle \alpha = -\pi : \pi')
>> ylabel('angle \beta = -\pi : \pi')
>> title('sin (\alpha^2+\beta^2)/(\alpha^2+\beta^2)')
```

Si une apostrophe est présente dans une chaîne de caractères, il faut la doubler car MATLAB l'utilise comme délimiteur de chaînes.

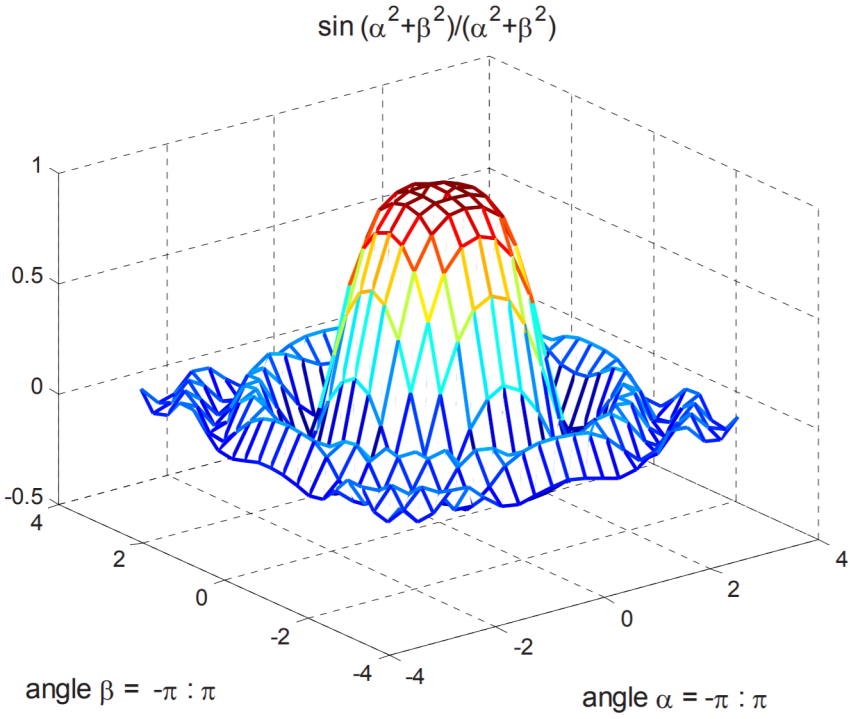


Figure 4 : Exemple de plot en 3D

10. Avant d'aller au Chapitre 2...

Exercice

[solution n°2 p.33]

Laquelle de ces écritures est correcte pour déclarer le vecteur y contenant les valeur 4, 6 et 7

- ☐ y=[4 6 7] ;
- ☐ y=[4 ;6 ;7] ;
- ☐ y=4,6,7

Exercice

[solution n°3 p.33]

Que signifie 'Command window'

- ○ Cette fenêtre résume toutes les commandes tapées dans le command window.
- ○ Contient des lignes qui te permettent à insérer tes commandes.
- ○ Cette fenêtre résume tous les variables qui ont été crées et utilisées dans votre programme.

Chapitre 2

1. Objectifs du chapitre 2

A la fin de ce chapitre vous serais capable de programmer sous fichier Script, de maîtriser les Instructions de contrôle ainsi que les opérateurs logiques

2. Les fichiers et la programmation avec MATLAB

Dans MATLAB, il y a deux types de fichiers : les fichiers de données et les fichiers de programmes (scripts et fonctions).

2.1. Fichiers de données

En plus des fichiers de données que l'on peut définir et utiliser par programmation, dans MATLAB on trouve les fichiers MAT. Ce sont des fichiers binaires (d'extension "*mat*") qui permettent de stocker et de restituer des variables utilisées dans l'espace de travail. Ces opérations sont réalisées respectivement par les commandes `save` et `load`.

Exemple : sauvegarde de variables

On définit une variable t.

```
>> t = [1 2 3]
t =
 1 2 3
```

On sauvegarde la variable t dans le fichier fic_t,

```
>> save fic_t t
```

Le fichier obtenu aura pour extension "mat" et sera sauvegardé sous le nom `fic_t.mat`.

Si on ne spécifie pas le nom d'une ou plusieurs variables dans les arguments de la commande `save`, toutes les variables de l'environnement seront sauvegardées.

Si on efface toutes les variables de la mémoire,

```
>> clear all
```

MATLAB ne connaît plus la variable t.

```
>> t
??? Undefined function or variable t.
```

Si l'on charge le fichier `fic_t,` la variable t est de nouveau présente dans l'espace de travail.

```
>> load fic_t
```

Cette vérification peut se faire, soit par l'appel de cette variable,

```
>> t
t =
 1 2 3
```

2.2. Fichiers de commandes et de fonctions

MATLAB peut exécuter une séquence d'instructions stockées dans un fichier. Ce fichier est appelé fichier M (M-file). Ce nom provient du fait que l'extension est "m".

La majorité de votre travail avec MATLAB sera liée à la manipulation de ces fichiers. Il y a deux types de fichiers M : les fichiers de commandes (fichiers scripts) et les fichiers de fonctions.

2.3. Les fichiers de commandes (scripts)

Un fichier de commandes ou script est une séquence d'instructions MATLAB. Les variables de ces fichiers sont locales à l'espace de travail.

Les valeurs des variables de votre environnement de travail peuvent être modifiées par les instructions des fichiers scripts.

Les fichiers de commandes (scripts) sont aussi utilisés pour la saisie de données. Dans le cas de grandes matrices, l'utilisation de scripts vous permet de corriger facilement et rapidement les erreurs de saisie.

Un fichier script peut appeler un autre ou s'appeler lui-même de façon récursive.

Nous proposons, dans ce qui suit, un exemple de script, stocké dans un fichier appelé courbe1.m, dont le code permet de tracer la courbe de la fonction y = x2 +5, sur l'intervalle [-5, 5].

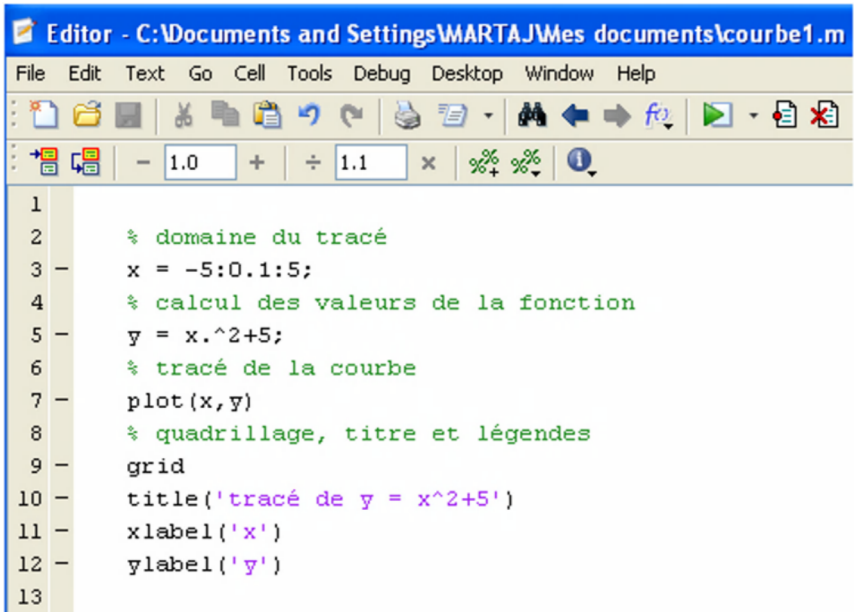


Figure 5 : Exemple d'un script pour tracer la courbe d'une fonction

Pour exécuter un script, dans la fenêtre de commande de MATLAB, il suffit de mettre son nom après le prompt ou de cliquer sur la flèche verte de l'éditeur ou bien la touche F5.

L'éditeur de la version R2015a de MATLAB permet de déboger le programme, facilite l'édition en affichant des couleurs différentes selon le type de données (commentaires en vert, chaînes de caractères en violet, etc.).

Ceci permet d'abord de sauvegarder le fichier puis de l'exécuter.

```
>> courbe1
```

L'exécution de ce script permet de tracer la courbe de parabole suivante :

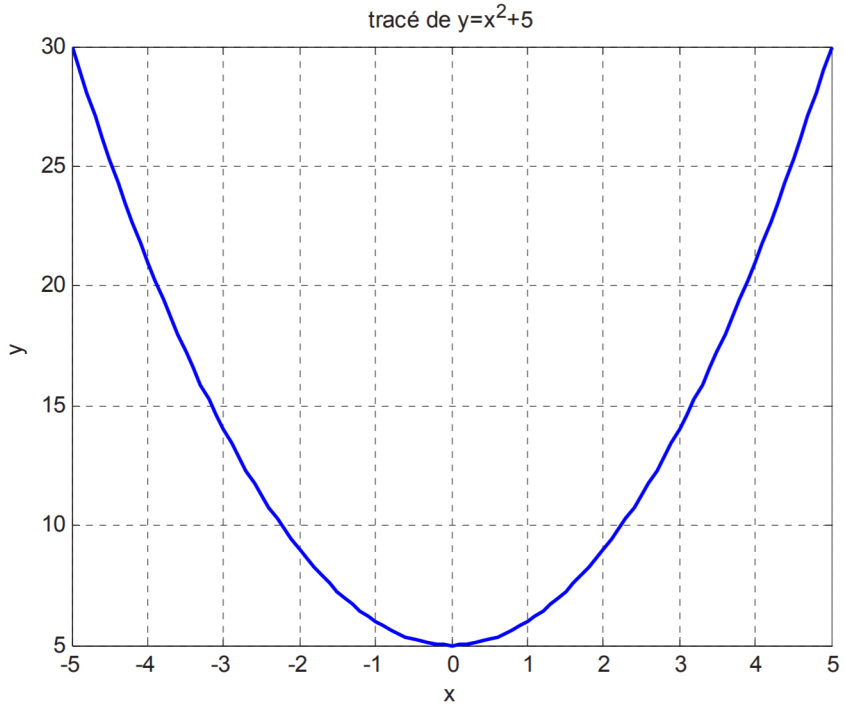


Figure 6 : Tracé de la courbe d'une fonction via un fichier sciprt

Pour exécuter ce script directement de l'éditeur, il suffit de cliquer sur la flèche verte dans le menu de l'éditeur ou bien la touche F5.

Remarque

Pour insérer des commentaires dans un programme, il suffit d'utiliser le symbole "%". Tout ce qui suit ce symbole, jusqu'à la fin de la ligne, est considéré comme un commentaire et n'est pas, par conséquent, interprété par MATLAB.

2.4. Les fichiers de fonctions

Les fichiers fonctions fournissent une extensibilité à MATLAB. Vous pouvez créer de nouvelles fonctions spécifiques à votre domaine de travail qui auront le même statut que toutes les autres fonctions MATLAB.

Les variables dans les fonctions sont par défaut locales, mais on peut définir des variables globales.

Exemple : Créer un fichier fonction

Nous allons écrire une fonction pour générer un tableau de n nombres aléatoires entiers compris entre 0 et une valeur maximale contenue dans une variable notée max.

```
1 function res = randint(n, max)
2 % res : tableau de n entiers compris entre 0 et max
3 % "rand" : génère un nombre aléatoire entre 0 et 1,
4 % "floor" : renvoie la partie entière d'un nombre
5 temp = rand(1,n);
6 res = floor((max+1).*temp);
```

Lorsqu'on sauvegarde ce programme, MATLAB propose de donner le même nom que cette fonction. Il est préférable de garder ce nom. Cet exemple sera ainsi stocké dans un fichier appelé `randint.m`.

La première ligne déclare le nom de la fonction (`randint.m`), les arguments d'entrée *(n, max)* et les arguments de sortie (valeurs retournées par la fonction, `res`). Sans cette première ligne, le fichier M correspond plutôt à un fichier script.

On peut remarquer que, contrairement aux langages classiques, les fonctions MATLAB peuvent donner en retour plusieurs arguments et de différents types.

Pour invoquer une fonction, il suffit de l'appeler suivant la syntaxe suivante :

```
resultat = nom_fonction(liste des arguments d'appel)
```

L'exemple suivant génère un vecteur aléatoire d'entiers, nommé "nb_alea", de longueur 10 et dont toutes les valeurs sont comprises entre 0 et 50.

```
1 >> nb_alea = randint(10,50)
2 nb_alea =
3 8 49 48 24 40 7 21 46 40 48
```

Remarque

Il est nécessaire de faire suivre la première ligne d'un fichier fonction par des lignes de commentaires dans lesquelles on décrira son but et ses arguments. Ces lignes seront utilisées par les commandes.

3. Instructions de contrôle

MATLAB dispose des instructions de contrôle suivantes : for, `while` et `if`.

La syntaxe de chacune de ces instructions est semblable à celles des langages classiques.

Il est important de savoir que beaucoup d'opérations nécessitent ces instructions dans des langages classiques tels que C et Fortran, alors que dans MATLAB, on peut s'en affranchir. Les opérations sur les matrices (addition, multiplication, etc.) sont les exemples les plus évidents.

3.1. L'instruction for

Syntaxe

```
for compteur = ValDébut : pas : ValFin

instructions

end
```

L'exemple suivant permet de générer, à l'aide de la boucle for, les carrés des n premiers entiers naturels.

```
1 % tableau des carrés des n premiers entiers naturels
2 n = 10;
3 x = [];
4 for i = 1:n
5  x = [x,i^2];
6 end
7 x
8
9 >> ncarres
10 x =
11  1 4 9 16 25 36 49 64 81 100
```

Ce programme peut être mis sur une même ligne lorsqu'on sépare les instructions par des par des virgules ou des points-virgules.

x = []; for i = 1:n, x = [x,i^2]; end

3.2. L'instruction while

Syntaxe

```
while conditions

instructions

end
```

Exemple

Dans l'exemple suivant, on affiche le plus petit entier naturel n tel que 2^n est supérieur ou égal à un nombre donné x.

```
x = 15; n = 0;
while 2^n < x
 n = n+1;
end
n
>> n_x
n =
 4
```

3.3. L'instruction if

Syntaxe

```
if conditions

instructions (si les conditions sont vérifiées)

elseif

else

instructions (si les conditions ne sont pas vérifiées)

end
```

Exemple

L'exemple suivant permet de vérifier si un entier naturel donné n est pair ou impair.

```
if rem(n,2) == 0
 disp('nombre pair')
else
 disp('nombre impair')
end
```

rem : retourne le reste de la division de deux nombres,

disp : affiche le message spécifié sous forme d'une chaîne de caractères.

4. Opérateurs relationnels et logiques

Des expressions relationnelles et logiques peuvent être utilisées dans MATLAB exactement comme dans les autres langages de programmation tels que le Fortran ou le C.

4.1. Opérateurs relationnels

Les opérateurs relationnels sont : <, <=, >, >=, ==, ~=

Ces opérateurs peuvent être utilisés avec des scalaires ou des matrices. Le résultat d'évaluation d'une expression relationnelle est 1 (vrai) ou 0 (faux).

Quand ces opérateurs sont appliqués à des matrices, le résultat est une matrice, de mêmes dimensions, formée de 0 et de 1, résultats de comparaisons élément à élément.

4.1.1. Opérateurs logiques

Les expressions relationnelles peuvent être combinées en utilisant les opérateurs logiques suivants :

&, |, ~ qui signifient respectivement "et" (AND), "ou" (OR) et "non" (NOT). Ces opérateurs sont appliqués sur les matrices élément par élément. Les opérateurs logiques ont une priorité plus faible que les opérateurs relationnels, qui à leur tour ont une priorité plus faible que les opérateurs arithmétiques.

Conseil

Il est conseillé d'utiliser les parenthèses afin d'éviter toute ambiguïté.

Prépares-vous ,un test!!!

IV

Exercice

[solution n°4 p.33]

A quoi sert les notations suivants : == ;~= ;& ;| ;~

- Afficher la liste de tous les variables utilisés contenant leurs types et sa taille mémoire.
- Ce sont égale, différent, et, ou et négation logique.
- Pour la saisie du variable v.

Exercice

[solution n°5 p.34]

i est un entier, Que sera affiché après l'exécution de ce programme:

```
For i:=1 To 3 Do
write(i,' ');
```

- 1 3 6
- 123
- 1 2 3

Conclusion

IV

Des notions de base sur la programmation sous MATLAB ont étés introduites. Le lecteur est bien invité de pratiquer et de exécuter les notions précédents.

Solutions des exercices

> **Solution** n°1

Exercice p. 7

C'est quoi MATLAB ?

- ☐ Un Langage de programmation
- ☐ Un système d'exploitation
- ☑ Un logiciel interactif de calcul matriciel

> **Solution** n°2

Exercice p. 23

Laquelle de ces écritures est correcte pour déclarer le vecteur y contenant les valeur 4, 6 et 7

- ☑ y=[4 6 7] ;
- ☑ y=[4 ;6 ;7] ;
- ☐ y=4,6,7

> **Solution** n°3

Exercice p. 23

Que signifie 'Command window'

- ○ Cette fenêtre résume toutes les commandes tapées dans le command window.
- ◉ Contient des lignes qui te permettent à insérer tes commandes.
- ○ Cette fenêtre résume tous les variables qui ont été crées et utilisées dans votre programme.

> **Solution** n°4

Exercice p. 31

A quoi sert les notations suivants : == ;~= ;& ;| ;~

- ○ Afficher la liste de tous les variables utilisés contenant leurs types et sa taille mémoire.
- ◉ Ce sont égale, différent, et, ou et négation logique.
- ○ Pour la saisie du variable v.

> **Solution** n°5

Exercice p. 31

i est un entier, Que sera affiché après l'exécution de ce programme:

```
For i:=1 To 3 Do
write(i,' ');
```

- ○ 1 3 6
- ○ 123
- ◉ 1 2 3

Abréviations

Matbab : Matrix Laboratory

Bibliographie

IV

Hunt, Brian R., Ronald L. Lipsman, and Jonathan M. Rosenberg. A guide to MATLAB: for beginners and experienced users. Cambridge University Press, 2014.

Martaj, Nadia, and Mohand Mokhtari. MATLAB R2009, SIMULINK et STATEFLOW pour Ingénieurs, Chercheurs et Etudiants. Springer Science & Business Media, 2010.