

# heig-vd

Haute Ecole d'Ingénierie et de Gestion  
du Canton de Vaud

Département Technologies Industrielles

## Matlab pour les ingénieurs

### Quelques exemples

i nstitut d'  
**A**utomatisation  
i ndustrielle



Prof. Freddy Mudry

# Introduction à Matlab :

## quelques exemples

### Table des matières

<b>1</b>	<b>Informations préliminaires</b>	<b>1</b>
1.1	Les bases de Matlab . . . . .	1
1.1.1	Donnée d'une variable réelle ou complexe . . . . .	1
1.1.2	Création de vecteurs et matrices . . . . .	2
1.1.3	Opérations sur les composantes ou les vecteurs . . . . .	3
1.1.4	Matrices particulières . . . . .	4
1.1.5	Aide en ligne . . . . .	4
1.2	Méthode de travail . . . . .	5
1.2.1	Les fichiers de commandes . . . . .	5
1.2.2	Les fichiers de fonctions . . . . .	5
1.2.3	Sauvegarde de données sur disque . . . . .	5
<b>2</b>	<b>Résolution d'un circuit électrique</b>	<b>6</b>
2.1	Description du circuit . . . . .	6
2.2	Calcul avec Matlab . . . . .	7
<b>3</b>	<b>Réponses d'un système linéaire</b>	<b>8</b>
3.1	Description du système . . . . .	8
3.2	Fichier Matlab . . . . .	8
<b>4</b>	<b>Mise en valeur de résultats expérimentaux</b>	<b>11</b>
4.1	Exemple 1 : caractéristique d'un ressort . . . . .	11
4.2	Exemple 2 : débit et température d'un flux d'air . . . . .	13
4.3	Exemple 3 : spectres du débit et de la température . . . . .	15
<b>5</b>	<b>Résolution d'une équation différentielle</b>	<b>16</b>
5.1	Équation du pendule simple . . . . .	16
5.2	Mise sous forme canonique . . . . .	16
5.3	Intégration numérique . . . . .	17
<b>6</b>	<b>Mouvement d'une charge électrique</b>	<b>19</b>
6.1	Équations fondamentales . . . . .	19
6.2	Description matricielle . . . . .	20
6.3	Description d'état d'un système linéaire . . . . .	20
6.4	Calcul de la trajectoire . . . . .	21

<b>7</b>	<b>Exercices</b>	<b>26</b>
7.1	Résolution d'un circuit électrique . . . . .	26
7.2	Réponses fréquentielle et temporelle d'un circuit . . . . .	26
7.3	Réponses fréquentielle et temporelle d'un filtre . . . . .	26
7.4	Portance d'une aile . . . . .	27
7.5	Modélisation de la caractéristique d'une diode . . . . .	28
7.6	Chute libre . . . . .	29
7.7	Saut à l'élastique . . . . .	30
7.8	Particule dans un champ électromagnétique . . . . .	30
<b>8</b>	<b>Liste de quelques fonctions Matlab</b>	<b>32</b>
8.1	Environnement Matlab . . . . .	32
8.1.1	Commandes et fonctions . . . . .	32
8.1.2	Informations sur l'espace de travail . . . . .	32
8.1.3	Commandes système . . . . .	32
8.1.4	Fenêtre de commandes . . . . .	33
8.1.5	Caractères spéciaux . . . . .	33
8.1.6	Opérateurs logiques . . . . .	33
8.1.7	Variables prédéfinies, durée, date . . . . .	34
8.1.8	Fonctions logiques . . . . .	34
8.1.9	Instructions de contrôle . . . . .	34
8.1.10	Instructions spécifiques . . . . .	35
8.2	Fonctions mathématiques . . . . .	35
8.2.1	Fonctions élémentaires . . . . .	35
8.2.2	Fonctions trigonométriques . . . . .	35
8.2.3	Fonctions prédéfinies . . . . .	36
8.3	Matrices et algèbre linéaire . . . . .	36
8.3.1	Opérateurs sur les matrices . . . . .	36
8.3.2	Opérateurs sur les composantes matricielles . . . . .	36
8.3.3	Manipulation des matrices . . . . .	36
8.3.4	Matrices prédéfinies . . . . .	37
8.3.5	Opérations sur les matrices . . . . .	37
8.3.6	Décomposition et factorisation de matrices . . . . .	37
8.4	Textes et chaînes de caractères . . . . .	38
8.5	Fonctions graphiques . . . . .	38
8.5.1	Graphiques 2D . . . . .	38
8.5.2	Annotation de graphiques . . . . .	38
8.5.3	Contrôle des fenêtres graphiques . . . . .	39
8.5.4	Sauvegarde et copie graphique . . . . .	39
8.5.5	Objets 3D . . . . .	39
8.5.6	Animations . . . . .	39
8.5.7	Apparence des graphiques . . . . .	39
8.5.8	Graphiques tridimensionnels . . . . .	40
8.5.9	Opérations sur les objets graphiques . . . . .	40
8.6	Opérations sur les polynômes . . . . .	40
8.7	Analyse de données et statistiques . . . . .	41
8.7.1	Analyse de données par colonne . . . . .	41
8.7.2	Analyse et traitement des signaux . . . . .	41
8.8	Intégration, interpolation et et dérivation numériques . . . . .	41

8.8.1	Intégration numérique . . . . .	41
8.8.2	Interpolation . . . . .	41
8.8.3	Différences finies . . . . .	42
8.9	Optimisation et équations non linéaires . . . . .	42
8.10	Modélisation et analyse de systèmes continus . . . . .	42
8.10.1	Construction d'un modèle . . . . .	42
8.10.2	Réponse temporelle . . . . .	42
8.10.3	Réponse fréquentielle . . . . .	42

# Introduction à Matlab :

## quelques exemples

Cette introduction à Matlab est proposée aux étudiants du département d'Électricité et Informatique de l'eivd avant le début de leur deuxième année de formation. Ce cours, donné pendant une journée, permet aux étudiants de travailler individuellement sur les exemples et exercices présentés ci-après.

Par ce recueil d'exemples, on souhaite montrer que l'usage de l'outil Matlab dans le domaine de l'ingénierie est simple et efficace et ainsi inciter nos étudiants à l'appliquer dans les cours et laboratoires qu'ils auront en deuxième et troisième année.

Les commandes Matlab sont présentées dans des situations réelles et suffisamment explicites pour que leur utilisation soit claire par elle-même. Les détails syntaxiques doivent être recherchés par les étudiants en recourant de manière intensive à l'aide en ligne Matlab.

## 1 Informations préliminaires

Matlab est un système interactif et convivial de calcul numérique et de visualisation graphique destiné aux ingénieurs et scientifiques. Il possède un langage de programmation puissant et simple à utiliser avec lequel l'utilisateur peut effectuer des calculs en ligne ou par l'intermédiaire d'un fichier de commandes [3], [4].

Le logiciel Matlab (*Matrix Laboratory*) est basé sur le calcul matriciel numérique. Tous les objets utilisés dans Matlab sont donc définis au travers de matrices ou vecteurs dont les valeurs sont, par définition, des grandeurs complexes. Il existe un très grand nombre d'opérateurs et fonctions distribués dans le logiciel de base et dans des boîtes à outils spécialisées. A ceci peut s'ajouter un outil de programmation graphique, Simulink, essentiel pour la simulation de systèmes dynamiques non linéaires.

L'environnement Matlab se présente sous la forme d'un espace de travail dans lequel un interpréteur de commandes exécute les opérations demandées.

### 1.1 Les bases de Matlab

#### 1.1.1 Donnée d'une variable réelle ou complexe

Avec Matlab, on définit sans autre des variables réelles ou complexes ; par exemple

```
aa = -1.234 ;           % nombre réel négatif
bb = 12/13 ;           % nombre réel positif
cc = 4.567 + j*8.765 ; % nombre complexe
dd = cc' ;             % son conjugué complexe
```

### 1.1.2 Création de vecteurs et matrices

Un vecteur ligne est introduit de la manière suivante :

```
vligne = [20, 6, 43, 66, 70, 8] ;
```

et un vecteur colonne comme suit :

```
vcol = [20 ; 6 ; 43 ; 66 ; 70 ; 8] ;
```

En l'absence de point-virgule (;), il s'affichera à l'écran sous la forme :

```
>> vligne =  
20  6  43  66  70  8
```

On peut changer son nom

```
v = vligne ;
```

On accède à la composante 4 en tapant :

```
v(4)  
>> ans =  
66
```

Un vecteur peut être retourné avec la commande `fliplr` :

```
y = fliplr(v)  
>> y=  
8  70  66  43  6  20
```

Une matrice peut être construite de différentes manières ;

soit

```
m = [ 2 4 6 8 ; 1 3 5 7 ; 11 13 17 19]
```

ou bien

```
m = [ 2  4  6  8  
      1  3  5  7  
      11 13 17 19]
```

ou bien

```
v1 = [ 2  4  6  8] ;  
v2 = [ 1  3  5  7] ;  
v3 = [11 13 17 19] ;  
m = [v1 ; v2 ; v3]
```

Ce qui, dans les 3 cas, donne sur l'écran

```
>> m =
      2   4   6   8
      1   3   5   7
     11  13  17  19
```

La deuxième colonne s'obtient en tapant

```
m2 = m( :,2)
>> m2 =
      4
      3
     13
```

L'affichage d'une sous-matrice est possible; par exemple

```
m3 = m(1 :3,2 :4)
>> m3 =
      4   6   8
      3   5   7
     13  17  19
```

Une matrice transposée se calcule avec l'apostrophe (') :

```
mt = m'
>> mt =
      2   1   11
      4   3   13
      6   5   17
      8   7   19
```

L'inverse d'une matrice se calcule aussi simplement ; par exemple

```
minv = inv(m3)
>> minv =
      6.0  -5.5  -0.5
     -8.5   7.0   1.0
      3.5  -2.5  -0.5
```

Dans Matlab, les indices des vecteurs et matrices doivent être des entiers positifs. L'indice 0 n'est donc pas admis.

### 1.1.3 Opérations sur les composantes ou les vecteurs

Avec Matlab, il faut être très attentif sur le type d'opérations souhaitées et cela en particulier lorsque l'on a affaire à des multiplications, des puissances, etc.

Comme exemple, considérons deux vecteurs que l'on multiplie entre eux :

```
v1 = [1 ; 2 ; 3] ;
v2 = [1, 2, 3] ;
```

L'opération  $v1*v2$  effectue le produit d'un vecteur colonne  $v1$  avec un vecteur ligne  $v2$  et le résultat est une matrice  $3 \times 3$  :

```
y1 = v1*v2
>> y1 =
     1     2     3
     2     4     6
     3     6     9
```

En croisant  $v1$  et  $v2$ , on obtient le produit scalaire :

```
y2 = v2*v1
>> y2 =
     14
```

Alors que si l'on veut effectuer le produit composante par composante, il faut utiliser l'opérateur de multiplication précédé d'un point ( $.*$ ) :

```
y3 = v1' .* v2
>> y3 =
     1     4     9
```

#### 1.1.4 Matrices particulières

Les fonctions `ones`, `zeros` permettent de construire des matrices remplies de 1 et de 0, respectivement. On a également la possibilité de construire une matrice identité (`eye`), diagonale (`diag`), une matrice dont les composantes sont aléatoires (`rand`), un vecteur dont les composantes sont espacées linéairement (`linspace`) ou logarithmiquement (`logspace`).

#### 1.1.5 Aide en ligne

À la moindre interrogation, n'hésitez pas à recourir à l'aide en ligne qui, généralement, est très bien documentée par des exemples et des renvois vers des fonctions complémentaires.

En tapant par exemple

```
help logspace
```

on obtient

```
LOGSPACE Logarithmically spaced vector.
LOGSPACE(d1, d2) generates a row vector of 50 logarithmically
equally spaced points between decades 10^d1 and 10^d2.  If d2
is pi, then the points are between 10^d1 and pi.

LOGSPACE(d1, d2, N) generates N points.

See also Linspace, :.
```



## 1.2 Méthode de travail

### 1.2.1 Les fichiers de commandes

Après quelques essais initiaux où les commandes sont passées en ligne, on éprouve très vite le besoin de ne pas perdre le fruit de son travail en écrivant les opérations, de plus en plus nombreuses et sophistiquées, dans un fichier de commandes d'extension `.m`.

Ce fichier peut être écrit avec votre éditeur de texte `ascii` (PFE, UltraEdit, etc.) ou celui fourni par Matlab. Une fois ce fichier sauvé dans votre répertoire de travail, n'oubliez pas d'indiquer son chemin d'accès à Matlab à l'aide de *File / Set Path*. L'exécution de la suite de commandes écrites dans le fichier `.m` se fera alors simplement en tapant son nom dans la fenêtre de commandes.

### 1.2.2 Les fichiers de fonctions

De nouvelles fonctions peuvent être créées et ajoutées à Matlab par l'utilisateur. Il suffit pour cela d'écrire un fichier `.m` dont le nom est obligatoirement le nom de la fonction utilisée par la suite.

Son entête doit avoir le format suivant :

```
function [arguments de sortie] = nom_de_fonction (arguments d'entrée)
```

**Exemple :** transformation des coordonnées rectangulaires en coordonnées polaires

```
function [module, argument] = rec2pol(x,y);
%   rec2pol(x,y) transforme les coordonnées rectangulaires (x,y)
%   en coordonnées polaires [module argument]
%   Remarque : l'argument est fourni en radians
%   Exemple d'utilisation :
%       [R A] = rec2pol(2,5);
%   Auteur : fmy / 26.10.2001

module = sqrt(x^2+y^2);
argument = angle(x + j*y);
```

Ces quelques lignes seront sauvées dans un fichier dont le nom sera `rec2pol.m` et placé dans un répertoire (`d :mesfonctions`, par exemple) contenant l'ensemble des fonctions que vous créez.

On notera que les variables internes à la fonction sont locales, contrairement à celles d'un fichier de commandes qui sont globales. De plus, les lignes précédées du symbole `%` (lignes de commentaires) serviront d'aide en ligne lorsque l'on tapera :

```
help rec2pol
```

### 1.2.3 Sauvegarde de données sur disque

Les variables définies dans l'espace de travail peuvent être sauvées dans un fichier `ascii` par la commande :

```
save chemin\nom_de_fichier.dat nom_de_variable -ascii
```

Un tel fichier `ascii` peut ensuite être relu soit par Matlab avec la fonction `load`, soit par d'autres programmes.

Le contenu de la fenêtre graphique courante peut être imprimé directement sur votre imprimante. On peut également le sauver dans un fichier pour être ensuite intégré dans un document PostScript

```
print chemin\nom_de_fichier.eps -depsc -tiff
```

ou Windows Meta File

```
print chemin\nom_de_fichier.wmf -meta
```

## 2 Résolution d'un circuit électrique

Étant donné le circuit de la figure 1, on souhaite calculer les valeurs efficaces et phases des courants lorsque

$$U_g = 220 [V], \quad f_g = 50 [Hz], \quad R_1 = 10 [\Omega]$$
$$R_2 = 3 [\Omega], \quad C = 10 [\mu F], \quad L = 100 [mH]$$

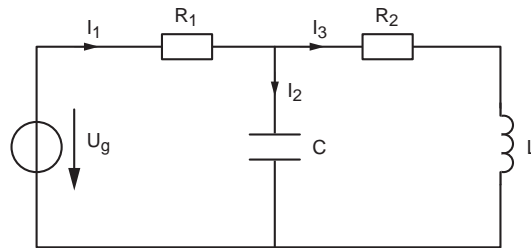


FIG. 1: Circuit électrique

### 2.1 Description du circuit

Pour résoudre ce circuit, il faut bien entendu commencer par écrire les équations qui le décrivent. Considérant les courants  $I_1$ ,  $I_2$ ,  $I_3$  circulant dans les 3 branches du circuit, celui-ci est complètement décrit par les équations suivantes :

$$U_g = R_1 I_1 + \frac{1}{j\omega C} I_2$$
$$0 = -\frac{1}{j\omega C} I_2 + (R_2 + j\omega L) I_3$$
$$0 = I_1 - I_2 - I_3$$

équations que l'on peut récrire sous forme matricielle :

$$\begin{pmatrix} R_1 & \frac{1}{j\omega C} & 0 \\ 0 & -\frac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

La solution s'obtient en multipliant à gauche les 2 membres de l'équation par l'inverse de la matrice décrivant le circuit :

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} R_1 & \frac{1}{j\omega C} & 0 \\ 0 & -\frac{1}{j\omega C} & R_2 + j\omega L \\ 1 & -1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} U_g \\ 0 \\ 0 \end{pmatrix}$$

## 2.2 Calcul avec Matlab

Le calcul des courants par Matlab en interprétant les lignes de commande d'un fichier (ici circuit.m) dont le contenu est :

```
% Calcul d'un circuit électrique
clear all; close all; format compact; format short g;
% donnees
fg = 50;      w = 2*pi*fg;  Ug = 220;  % Ug = 220 Veff
R1 = 10;      R2 = 3;      L = 100e-3;  C = 10e-6;
% rappel des equations du circuit
% R1 I1 + 1/jwC I2 +      0 I3 = Ug
% 0 I1 - 1/jwC I2 + (R2+jwL) I3 = 0
% I1 -      I2 -      I3 = 0
% description matricielle du circuit
Z = [ R1      +1/(j*w*C)      0
      0      -1/(j*w*C)      R2+j*w*L
      1      -1      -1];
U = [Ug; 0; 0];      % vecteur colonne
% resolution du circuit
I = inv(Z) * U;
% affichage des valeurs
I
Ieff = abs(I)
PhaseI = angle(I) * 180 / pi
% affichage sous forme matricielle
Courants = [I Ieff PhaseI]
```

On lance l'exécution de ces commandes en tapant simplement le nom du fichier sans extension

```
>> circuit
I =
    2.1521 - 5.4723i
   -0.1719 + 0.6235i
    2.3240 - 6.0959i
Ieff =
    5.8803
    0.6468
    6.5238
PhaseI =
   -68.5319
```

```

105.4142
-69.1310
Courants =
1.0e+002 *
    0.0215 - 0.0547i    0.0588    -0.6853
   -0.0017 + 0.0062i    0.0065     1.0541
    0.0232 - 0.0610i    0.0652    -0.6913

```

### 3 Réponses d'un système linéaire

#### 3.1 Description du système

On considère ici un filtre RL-C passe-bas décrit par sa fonction de transfert :

$$H(s) = \frac{1}{1 + (s/\omega_0)/Q_0 + (s/\omega_0)^2}$$

avec

$$\omega_0 = \frac{1}{\sqrt{LC}}, \quad Q_0 = \frac{\omega_0 L}{R}$$

Le calcul et traçage des réponses fréquentielle (fig. 2a) et temporelle (fig. 2b) se fait tout naturellement à l'aide de Matlab comme le montre le fichier ci-dessous.

#### 3.2 Fichier Matlab

```

% calcul des réponses d'un système linéaire décrit
% par une fonction de transfert d'ordre 2
%
%          1
% H(s) = -----
%          (s/w0)^2 + (s/w0)/Q0 + 1

clear all; close all; clc;
format compact; format short g;

% données : circuit LC-R (passe-bas)
R = 500; L = 1e-3; C = 1e-9;
w0 = 1/sqrt(L*C), Q0 = w0*L/R
num = [0, 0, 1];
den = [1/w0^2, 1/Q0/w0, 1];

% calcul de la réponse fréquentielle
f0 = w0/2/pi;
ech_log = -2 :0.01 :+2;
ff = f0*10.^ech_log;
jw = j*2*pi*ff;
num_jw = num(1)*jw.^2 + num(2)*jw + num(3);
den_jw = den(1)*jw.^2 + den(2)*jw + den(3);
Hjw = num_jw./den_jw;
Hdb = 20*log10(abs(Hjw));
phi = angle(Hjw)/pi*180;

```

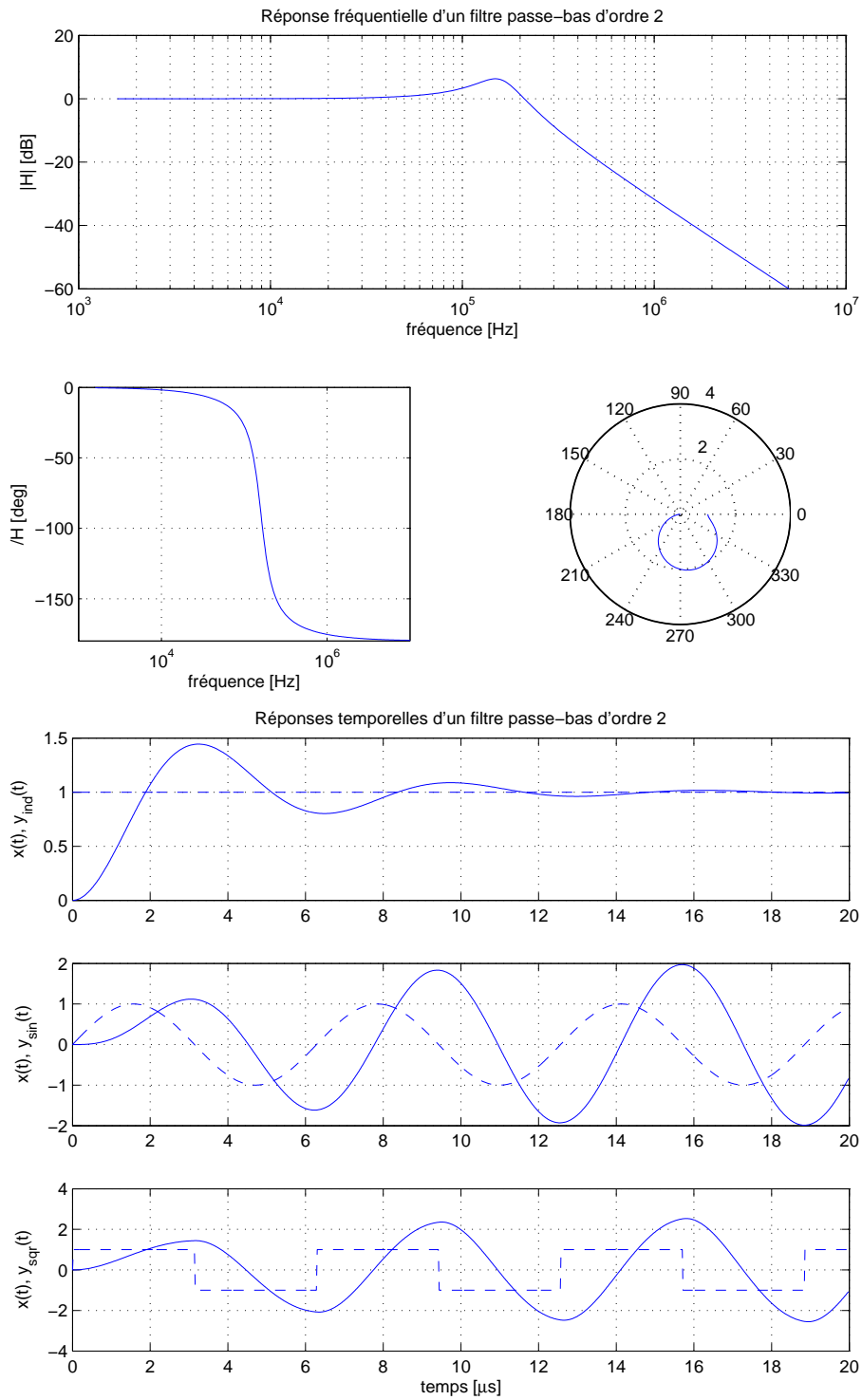


FIG. 2: a) Diagrammes de Bode et polaire de la réponse fréquentielle  
 b) Trois réponses temporelles

```

% calcul de trois réponses temporelles
tmax = 10*L/R; Npts = 1000; dt = tmax/Npts;
tt = 0 :dt :tmax-dt;
xti = ones(size(tt));
yti = lsim(num, den, xti, tt);
xts = sin(2*pi*tt*f0);
yts = lsim(num, den, xts, tt);
xtc = sign(sin(2*pi*tt*f0));
ytc = lsim(num, den, xtc, tt);

% traçage de la réponse fréquentielle
figure;
subplot(2,1,1);
    semilogx(ff, Hdb);
    grid on;
    axis([1e3, 1e7, -60, +20]);
    xlabel('fréquence [Hz]');
    ylabel('|H| [dB]');
    title('Réponse fréquentielle d\'un filtre passe-bas d\'ordre 2')
subplot(2,2,3);
    semilogx(ff, phi);
    grid on;
    axis([1e3,1e7,-180,0]);
    xlabel('fréquence [Hz]');
    ylabel('/H [deg]');
subplot(2,2,4);
    polar(angle(Hjw), abs(Hjw));
print -depsc rep_freq_rlc.eps

% tracage des réponses temporelles
figure;
subplot(3,1,1);
    plot(tt*1e6, xti,'--');
    grid on; hold on;
    plot(tt*1e6, yti);
    axis([0, 20, 0, 1.5]);
    ylabel('x(t), y_{ind}(t)');
    title('Réponses temporelles d\'un filtre passe-bas d\'ordre 2')
subplot(3,1,2);
    plot(tt*1e6, xts,'--');
    grid on; hold on;
    plot(tt*1e6, yts);
    ylabel('x(t), y_{sin}(t)');
subplot(3,1,3);
    plot(tt*1e6, xtc,'--');
    grid on; hold on;
    plot(tt*1e6, ytc);
    xlabel('temps [\mus]');
    ylabel('x(t), y_{sqr}(t)');
print -depsc rep_temp_rlc.eps

```

## 4 Mise en valeur de résultats expérimentaux

### 4.1 Exemple 1 : caractéristique d'un ressort

On considère ici un ressort que l'on tend plus ou moins tout en mesurant la force qui lui est appliquée à l'aide d'un dynamomètre dont la précision est d'environ 0.5 [N]. Les résultats que l'on a obtenus sont les suivants :

Longueur [cm]	4.2	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0
Force [N] +1/ -0.5 [N]	0.0	1.1	2.0	3.2	3.9	4.6	5.8	7.0	8.3	9.0	9.5

Dans ce qui suit, on souhaite :

1. Tracer le graphe de la force en fonction de la longueur avec les barres d'erreurs.
2. Mettre en valeur le graphe à l'aide d'un titre et d'informations portées sur l'abscisse et l'ordonnée.
3. Rechercher une loi polynomiale représentant au mieux cette caractéristique ; celle d'un ressort est généralement linéaire, éventuellement cubique.
4. Mesurer la qualité des modèles proposés pour représenter le ressort.
5. Afficher les informations sur le graphe lui-même.

Un programme réalisant le cahier des charges demandé peut être le suivant :

```
% Exemple de traitement des donnees
clear all ; close all ; format compact ; format short g ;

% elongation d'un ressort : valeurs mesurees
x = [4.2 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0] ; % [cm]
force = [0.0 1.1 2.0 3.2 3.9 4.6 5.8 7.0 8.1 9.0 9.5] ; % [N]

% vecteurs des erreurs
ErrNeg = -0.5*ones(size(force)) ;
ErrPos = +1.0*ones(size(force)) ;

% graphes des mesures
plot(x,force,'o') ;

% regressions lineaire et cubique
coeff1 = polyfit(x,force, 1) % coeff. du polynôme d'ordre 1
coeff3 = polyfit(x,force, 3) % coeff. du polynôme d'ordre 3

% calcul des graphes des modeles d'ordre 1 et 3
xfit = linspace(0,20,201) ; % abscisse plus fine (200 points)
F1 = polyval(coeff1,xfit) ; % valeurs du polynôme d'ordre 1
F3 = polyval(coeff3,xfit) ; % valeurs du polynôme d'ordre 3

% tracage des mesures avec erreurs et des modeles
errorbar(x,force,ErrNeg, ErrPos,'o') ;
```

```

hold on; grid on; % maintien du graphe et pose d'une grille
plot (xfit,F1, xfit,F3,'--');
axis ([0 20 -5 13]);
title('Force d'un ressort');
xlabel('x [cm]'); ylabel('F(x) [N]');

% affichage des infos
texte = ['F_1(x) = ', num2str(coeff1(1),3), ' x + '];
texte = [texte, num2str(coeff1(2),3)];
text(1,11, texte);
texte = ['F_3(x) = ', poly2str(coeff3,'x')];
text(2.1,-3, texte);

% ecart type = sigma = mesure de la dispersion des ecart
F1 = polyval(coeff1, x); % valeurs du modele d'ordre 1
ecart1 = force - F1;
eqm1 = sum(ecart1 .^ 2)/length(ecart1);
sigma1 = sqrt(eqm1);
text (1,10.2, ['\sigma_1 = ', num2str(sigma1,3), ' [N]']);

F3 = polyval(coeff3, x); % valeurs du modele d'ordre 3
ecart3 = force - F3;
eqm3 = sum(ecart3 .^ 2)/length(ecart3);
sigma3 = sqrt(eqm3);
text (2.1,-3.8, ['\sigma_3 = ', num2str(sigma3,3), ' [N]']);
print -deps ressort.eps

```

Les résultats sont présentés dans la figure 3.

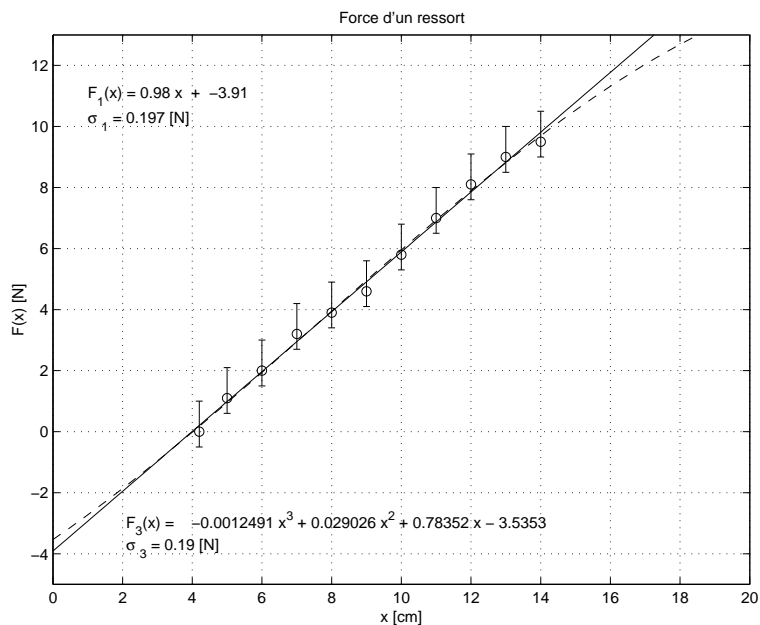


FIG. 3: Force d'un ressort et ses modèles d'ordre 1 et 3



## 4.2 Exemple 2 : débit et température d'un flux d'air

Dans cet exemple, on se propose de montrer comment on lit des valeurs fournies par un fichier `ascii` et comment on tire quelques informations utiles avant de les porter sur un graphe (figure 4).

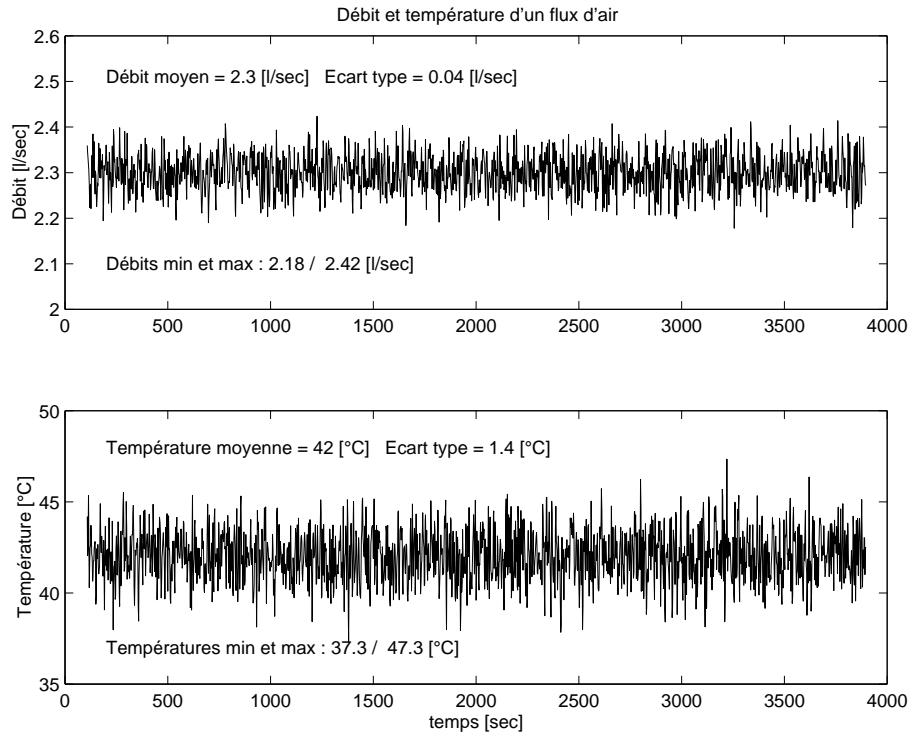


FIG. 4: Débit et température d'un flux d'air

Lors d'une expérience, on a enregistré l'évolution du débit et de la température d'un flux d'air au cours du temps. Les valeurs numériques ont été portées dans un fichier `air.dat` comportant 3 colonnes :

- le temps exprimé en secondes
- le débit donné en litres par seconde
- la température mesurée en degrés Celsius :

```
1.080000e+002  2.3597495e+000  4.4197579e+001
1.100000e+002  2.3446581e+000  4.2045259e+001
1.120000e+002  2.3312441e+000  4.2574619e+001
.....
3.894000e+003  2.2748863e+000  4.2438009e+001
3.896000e+003  2.2713996e+000  4.2516874e+001
```

Voici le contenu du programme permettant de lire le fichier de données et de construire la figure 4 :

```
% mesures de debit et temperature
clear all; close all; format compact; format short g;

% chargement des mesures
```

```

mesures = load('air.dat');
temps = mesures(:,1);
debit = mesures(:,2);
degre = mesures(:,3);
% pre-visualisation des mesures
subplot(2,1,1); plot(temps, debit);
subplot(2,1,2); plot(temps, degres);

% recherche des valeurs min et max
Qmin = min(debit), Qmax = max(debit)
Tmin = min(degre), Tmax = max(degre)

% calcul des valeurs moyennes
DebitMoyen = mean(debit)
DegreMoyen = mean(degre)

% mesure de la dispersion des valeurs
% std = ecart type = deviation standard
BruitDebit = std(debit-DebitMoyen)
BruitDegre = std(degre-DegreMoyen)

% tracage du debit
figure;
subplot(2,1,1); plot(temps,debit);
axis([0 4000 2 2.6]);
title('Débit et température d'un flux d'air');
ylabel('Débit [l/sec]');

% informations pour le graphe
texte1 = ['Débit moyen = ', num2str(DebitMoyen,2), ' [l/sec]'];
texte2 = ['Ecart type = ', num2str(BruitDebit,2), ' [l/sec]'];
text(200,2.51,[texte1 ' ' texte2]);
texte = ['Débits min et max : ', num2str(Qmin,3)];
texte = [texte ' / ', num2str(Qmax,3), ' [l/sec]'];
text(200,2.1, texte);

% tracage de la temperature
subplot(2,1,2); plot(temps,degre);
axis([0 4000 35 50]);
xlabel('temps [sec]');
ylabel('Température [°C]');

% information pour le graphe
texte1 = ['Température moyenne = ', num2str(DegreMoyen,2), ' [°C]'];
texte2 = ['Ecart type = ', num2str(BruitDegre,2), ' [°C]'];
text(200,48,[texte1 ' ' texte2]);
texte = ['Températures min et max : ', num2str(Tmin,3)];
texte = [texte ' / ', num2str(Tmax,3), ' [°C]'];
text(200,37, texte);

```

```
print -deps airtemp.eps
```

### 4.3 Exemple 3 : spectres du débit et de la température

Le but de cet exemple est de montrer comment on peut calculer et afficher le spectre des signaux précédents. Cependant, il est important de bien réaliser que, derrière ces calculs, c'est une partie importante du traitement numérique des signaux qui est appliquée et que ce n'est pas le but de la présenter ici.

L'analyse temporelle faite dans le paragraphe précédent se poursuit dans l'espace des fréquences à l'aide des commandes ci-après.

```
% preparation pour l'analyse spectrale
% la longueur du signal analyse doit etre une puissance de 2
% inferieure ou egale a la longueur du signal original
puissance = floor(log(length(debit))/log(2))
kmax = 2^puissance;
debit2 = debit(1 :kmax);
degre2 = degre(1 :kmax);

% suppression de la valeur moyenne
debit2 = debit2-mean(debit2);
degre2 = degre2-mean(degre2);

% transformation de Fourier des signaux a valeur moyenne nulle
Qf = fft(debit2);
Tf = fft(degre2);

% relations temps-frequence
dt = temps(2)-temps(1);
duree = kmax*dt;
fmin = 0; fmax = 1/dt; df = 1/duree;
ff = fmin :df :fmax;

% preparation pour le tracage
Qfdb = 20*log10(abs(Qf));
Tfdb = 20*log10(abs(Tf));
ff = ff(1 :kmax/2); % le spectre se repete au dela de kmax/2
Qfdb = Qfdb(1 :kmax/2);
Tfdb = Tfdb(1 :kmax/2);

% recherche d'un pic spectral eventuel sur le debit
[pic kplic]= max(Qfdb);
fpic = kplic*df;
texte = ['f_{pic} = ', num2str(fpic,2), ' [Hz]'];

% tracage
figure;
subplot(2,1,1); plot(ff,Qfdb);
axis([0 0.1 -20 40]); grid;
text(1.1*fpic, 1.1*pic,texte);
ylabel('Q(f) [dB]');
title('Spectres du débit et de la température');
subplot(2,1,2); plot(ff,Tfdb);
axis([0 0.1 0 60]); grid;
```

```
xlabel('fréquence [Hz]'); ylabel('T(f) [dB]');
print -deps airfreq.eps
```

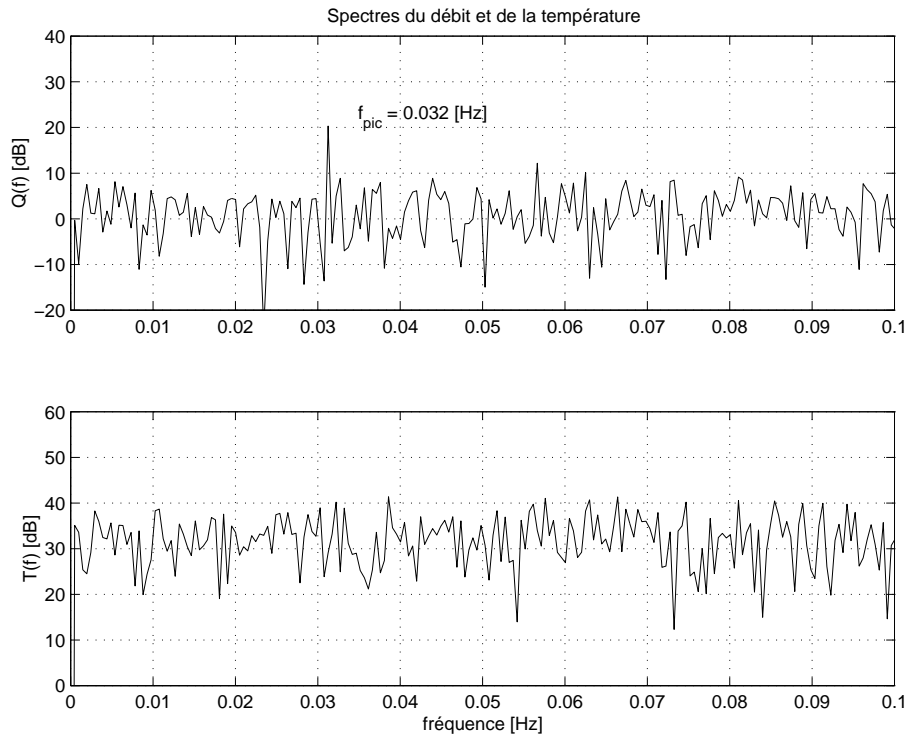


FIG. 5: Spectres du débit et de la température

L'analyse spectrale complétée par son graphe (figure 5) montre qu'il existe une raie spectrale sur le débit située en  $f = f_{pic} = 0.032 [Hz]$ . Cela indique la présence d'une perturbation périodique de période  $1/f_{pic} \simeq 30 [sec]$  qui n'était pas du tout perceptible dans le graphe temporel.

## 5 Résolution d'une équation différentielle

### 5.1 Équation du pendule simple

La mise en équation du pendule simple est traitée dans le cours de physique [5]. On y montre que son mouvement est décrit par une équation différentielle non-linéaire d'ordre 2 :

$$\frac{d^2\theta(t)}{dt^2} = \ddot{\theta}(t) = -\frac{g}{L} \sin(\theta(t))$$

Dans le cas où l'angle  $\theta(t)$  est petit, cette équation peut être approchée par une équation linéaire qui est :

$$\frac{d^2\theta(t)}{dt^2} = \ddot{\theta}(t) = -\frac{g}{L} \theta(t)$$

### 5.2 Mise sous forme canonique

Avant de résoudre numériquement un système décrit par une équation différentielle, il faut remplacer cette équation d'ordre  $n$  par  $n$  équations différentielles d'ordre 1 dont les

variables  $x_1(t), \dots, x_n(t)$  sont la variable originale et ses dérivées successives. Dans notre cas, cela donne :

$$\begin{aligned}x_1(t) &\equiv \theta(t) \\x_2(t) &\equiv \dot{\theta}(t)\end{aligned}$$

Le système est alors complètement décrit par :

$$\begin{aligned}\frac{dx_1(t)}{dt} &\equiv \dot{\theta}(t) = x_2(t) \\ \frac{dx_2(t)}{dt} &\equiv \ddot{\theta}(t) = -\frac{g}{L} \sin(x_1(t))\end{aligned}$$

Sous Matlab, la description du système doit se faire dans un fichier `*.m` décrivant la fonction qui fournit les  $n$  dérivées sous la forme d'un vecteur colonne. Dans notre cas, cela donne :

*fichier EDpendule.m :*

```
% Equation differentielle d'un pendule simple
function dx = EDpendule(t, x);
    g = 9.81;
    longueur = 1.0;
    theta = x(1);
    dtheta = x(2);
    ddtheta = -g/longueur*sin(theta);
    dx = [dtheta; ddtheta];
```

Dans une approche plus générale, les paramètres ( $g$  et  $L$ ) du pendule peuvent être définis dans le programme principal et passés en arguments à la fonction décrivant l'équation différentielle. De plus, sachant que le vecteur  $x$  contient les variables  $\theta$  et  $\dot{\theta}$ , l'écriture de la fonction peut se faire dans une forme plus compacte (mais peut-être moins évidente à comprendre).

L'illustration en est donnée dans le fichier suivant écrit pour l'approximation linéaire du pendule :

*fichier EDpendlin.m :*

```
% Equation differentielle lineaire
function dx = EDpendlin(t, x, options, param);
    g = param(1);
    longueur = param(2);
    dx1 = x(2);
    dx2 = -g/longueur*x(1);
    dx = [dx1; dx2];
```

### 5.3 Intégration numérique

L'intégration numérique d'un système différentiel se fait à l'aide de divers algorithmes d'intégration qui sont fournis par Matlab. L'algorithme le plus fréquemment utilisé est celui de Runge-Kutta qui, dans Matlab, est désigné sous le nom de `ode45` (Ordinary Differential Equation, approximation d'ordre 4 et 5).

A l'appel de `ode45`, il faut donner le nom du fichier contenant l'équation différentielle, le domaine temporel désiré pour la résolution et les conditions initiales. On peut, si on le souhaite, modifier les options de résolution et fournir des paramètres :

```
ode45('fichierED.m',[tmin tmax],CI,options,param)
```

La résolution du pendule simple est donnée ci-dessous. Dans un but de comparaison, on y a ajouté la solution linéaire qui est tracée en traitillé dans la figure 6.

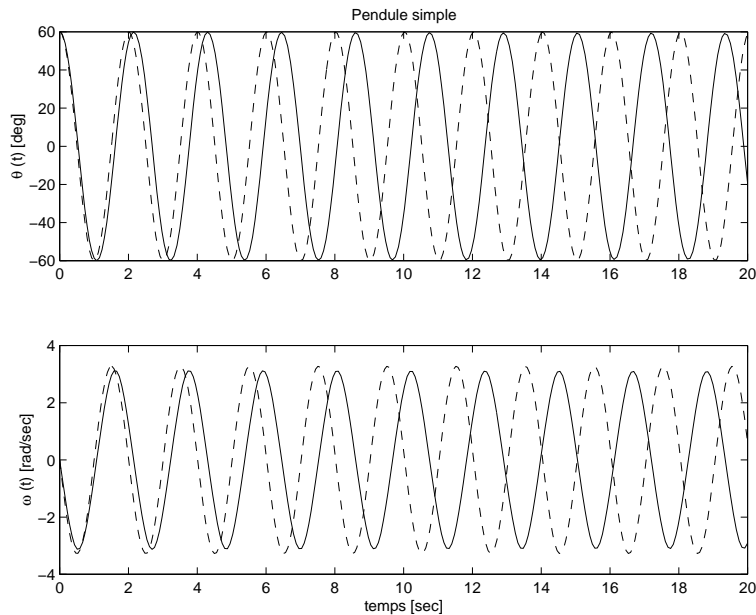


FIG. 6: Évolution d'un pendule simple

```
% Analyse d'un pendule simple
clear all; close all; format compact; format short g;

% conditions initiales
pos0 = 60*pi/180; % radians
vit0 = 0;

% domaine temporel
t0 = 0; tmax = 20;

% resolution numerique de l'equation non lineaire
% sans passage de parametres
[tt1, Xs1] = ode45('EDpendule.m',[t0 tmax],[pos0 vit0]);
pos1 = Xs1(:,1)/pi*180; % degrees
vit1 = Xs1(:,2);

% parametres
g = 9.81; longueur = 1;
param(1) = g; param(2) = longueur;

% resolution numerique de l'equation lineaire
```

```

% avec passage de parametres
[tt2, Xs2] = ode45('EDpendlin.m',[t0 tmax],[pos0 vit0],[],param);
pos2 = Xs2(:,1)/pi*180; % degres
vit2 = Xs2(:,2);

% tracage
figure;
subplot(2,1,1); plot(tt1,pos1,tt2,pos2,'--');
title('Pendule simple');
ylabel('\theta (t) [deg]');
subplot(2,1,2); plot(tt1,vit1,tt2,vit2,'--');
ylabel('\omega (t) [rad/sec]');
xlabel('temps [sec]');
print -deps pendule.eps

```

## 6 Mouvement d'une charge électrique

### 6.1 Équations fondamentales

On considère ici le mouvement d'une charge  $q$  placée dans des champs électrique  $\vec{E}$  et magnétique  $\vec{B}$  constants et uniformes (figure 7).

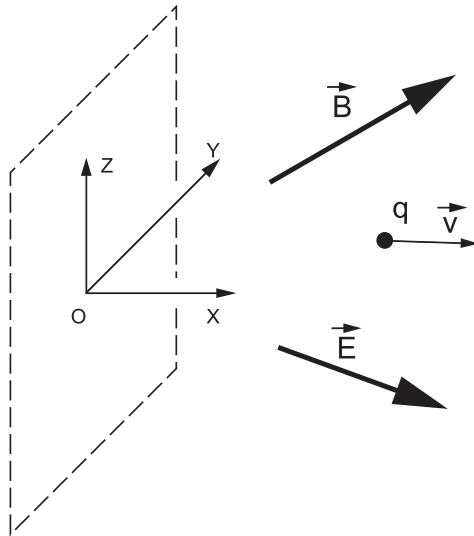


FIG. 7: Charge dans un champ électromagnétique

Les lois de Newton et de l'électromagnétisme [5] permettent alors de décrire le mouvement de la charge  $q$  de masse  $m$  :

$$\vec{F} = m \vec{a} = m \dot{\vec{v}} = q \vec{E} + q \vec{v} \times \vec{B}$$

Cette équation vectorielle s'écrit également sous la forme :

$$m \begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix} = q \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} + q \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \times \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$$

Effectuant le produit vectoriel, cela donne :

$$m \begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix} = q \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} + q \begin{pmatrix} v_y B_z - v_z B_y \\ v_z B_x - v_x B_z \\ v_x B_y - v_y B_x \end{pmatrix}$$

On voit ainsi que le mouvement de la charge est décrit par 3 équations différentielles couplées d'ordre 1.

## 6.2 Description matricielle

Définissant un vecteur d'état  $\vec{X}_s$  et sa dérivée  $\dot{\vec{X}}_s$

$$\vec{X}_s = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \quad \dot{\vec{X}}_s = \begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix}$$

l'équation du mouvement s'écrit :

$$\dot{\vec{X}}_s = \begin{pmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{pmatrix} = \frac{q}{m} \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} + \frac{q}{m} \begin{pmatrix} X_s(2) B_z - X_s(3) B_y \\ X_s(3) B_x - X_s(1) B_z \\ X_s(1) B_y - X_s(2) B_x \end{pmatrix}$$

Ce qui, en terme d'équation matricielle, devient :

$$\dot{\vec{X}}_s = \begin{pmatrix} \dot{X}_s(1) \\ \dot{X}_s(2) \\ \dot{X}_s(3) \end{pmatrix} = \begin{pmatrix} 0 & +\frac{q}{m} B_z & -\frac{q}{m} B_y \\ -\frac{q}{m} B_z & 0 & +\frac{q}{m} B_x \\ +\frac{q}{m} B_y & -\frac{q}{m} B_x & 0 \end{pmatrix} \begin{pmatrix} X_s(1) \\ X_s(2) \\ X_s(3) \end{pmatrix} + \begin{pmatrix} \frac{q}{m} E_x \\ \frac{q}{m} E_y \\ \frac{q}{m} E_z \end{pmatrix} \cdot 1$$

Une approche analytique du problème permet de montrer que la particule oscille sinusoidalement avec une pulsation proportionnelle à l'amplitude du champ magnétique :

$$\omega_n = \frac{q}{m} B$$

## 6.3 Description d'état d'un système linéaire

La description matricielle ci-dessus porte le nom de représentation d'état d'un système et correspond à l'écriture conventionnelle générale suivante :

$$\begin{aligned} \dot{\vec{X}}_s &= A \vec{X}_s + B u \\ \vec{Y} &= C \vec{X}_s + D u \end{aligned}$$

dans laquelle :

- $\vec{X}_s$  est le vecteur d'état du système,
- $\vec{Y}$  sa sortie,
- $A$  la matrice d'état décrivant le système,



- $B$  le vecteur de commande,
- $u$  le signal de commande,
- $C$  et  $D$  les matrices décrivant la sortie  $\vec{Y}$ .

Dans le cas de notre problème, les dimensions des vecteurs et matrices sont  $X_s(3, 1)$ ,  $A(3, 3)$ ,  $B(3, 1)$  et  $u(1, 1)$ . Admettant que le vecteur de sortie  $\vec{Y}$  est constitué des 3 composantes de la vitesse, on a évidemment  $\vec{Y} = \vec{X}_s$ . La matrice  $C(3, 3)$  est alors une matrice unité de dimension 3 et le vecteur  $D(3, 1)$  est nul.

## 6.4 Calcul de la trajectoire

Ce système étant linéaire, le calcul de l'évolution temporelle peut se faire sans intégration numérique. Ce qui, dans Matlab, se fait de la manière suivante :

```
% Mouvement d'une charge dans un champ electromagnetique
clear all; close all; format compact;

% constantes
q = 1.602e-19;
m = 9.109e-31;
Bx = 0.0; By = 0.01; Bz = 0.0;
Ex = 1.0; Ey = 0.0; Ez = 0.0;

% description d'etat :
A = q/m * [ 0    +Bz  -By
            -Bz   0    +Bx
            +By  -Bx   0] ;
B = q/m*[Ex; Ey; Ez] ;

% matrices de sortie :
C = [1 0 0
     0 1 0
     0 0 1] ;
D = zeros(3,1) ;

% representation d'etat du systeme
sys = ss(A,B,C,D) ;
```

```

% calcul de la reponse indicielle (=> u = 1) :
tmax = 1e-8; kmax = 1000 ;
dt = tmax/kmax ;
tt = 0 :dt :tmax ;
y = step(sys,tt) ;

% extraction des 3 composantes de la vitesse
Vx = y( :,1) ; Vy = y( :,2) ; Vz = y( :,3) ;

```

Comme le vecteur de sortie  $\vec{Y}$  contient les 3 composantes de la vitesse de la particule et que l'on désire connaître sa trajectoire, il est possible de passer des vitesses aux positions par intégration numérique. Ce qui donne pour la coordonnée  $x(t)$  :

$$x(t) = \int_0^t v_x(t) dt \simeq x[n] = \sum_{k=0}^n v_x[k] \Delta t \quad n = 0 \cdots N_{points}$$

Dans Matlab, cette somme cumulative est réalisée par la commande `cumsum` (on notera qu'il existe également la fonction `cumtrapz` qui réalise une intégration trapézoïdale plus précise) :

```

% calcul des positions (integration d'ordre 0 => cumsum)
xt = dt*cumsum(Vx) ;
yt = dt*cumsum(Vy) ;
zt = dt*cumsum(Vz) ;

% tracage des positions
figure ;
subplot (3,1,1) ; plot (tt,xt) ;
title('Déplacement de la particule') ; ylabel('x(t)') ;
subplot (3,1,2) ; plot (tt,yt) ; ylabel('y(t)') ;
subplot (3,1,3) ; plot (tt,zt) ;
ylabel('z(t)') ; xlabel('temps [sec]') ;

```

Les résultats obtenus sont présentés dans les figures 8 et 9. La visualisation 3D de la trajectoire est réalisée simplement avec les commandes

```

plot3(xt,yt,zt) ; axis square ; grid on ;

```

On peut encore ajouter les vecteurs  $E_x$  et  $B_y$  avec la commande `quiver3(x,y,z, u,v,w, 0, 'x')`. Pour plus d'informations, on consultera avantageusement l'aide Matlab.

Les commandes ayant servi au traçage de la figure 9 sont données ci-dessous sans plus de commentaires :

```
figure ;
plot3(xt,yt,zt) ; % tracage 3D
axis square ; grid ; hold on ;
axis([-5e-8,15e-8,-1,1,0,1e-6]) ; % choix des axes
h1=quiver3(5e-8,0,0, -5e-8,0,0, 0, 'v') ; % vecteur E = Ex
set(h1, 'LineWidth', 2) ; % vecteur en trait gras
text(3.5e-8, -0.15, 0, 'E_x') ; % legende pour E
h2=quiver3(0,0.5,0, 0, -0.5, 0, 0, 'v') ; % vecteur B = By
set(h2, 'LineWidth', 2) ; % vecteur en trait gras
text(-2e-8, 0.4, 0, 'B_y') ; % legende pour B
title('Trajectoire de la particule') ;
print -deps qtraject.eps
```

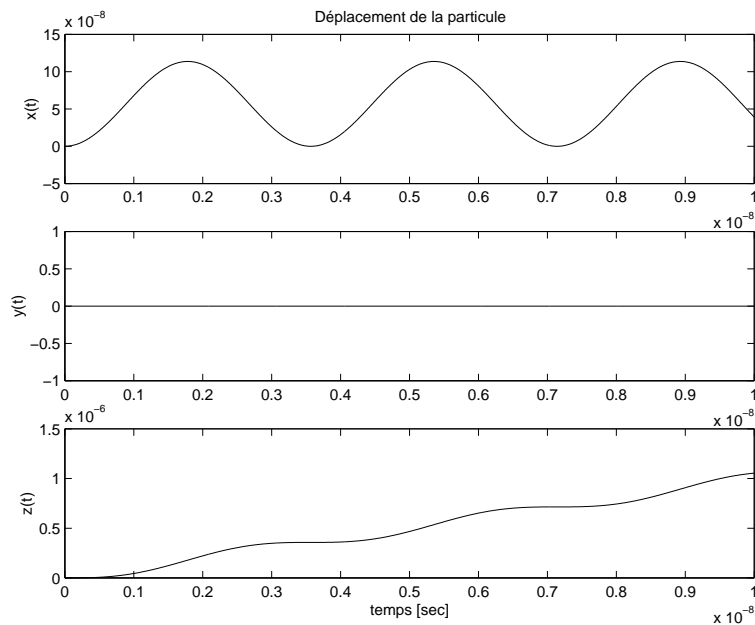


FIG. 8: Coordonnées d'une particule chargée dans un champ électrique  $\vec{E} = (E_x; 0; 0)$  et magnétique  $\vec{B} = (0; B_y; 0)$

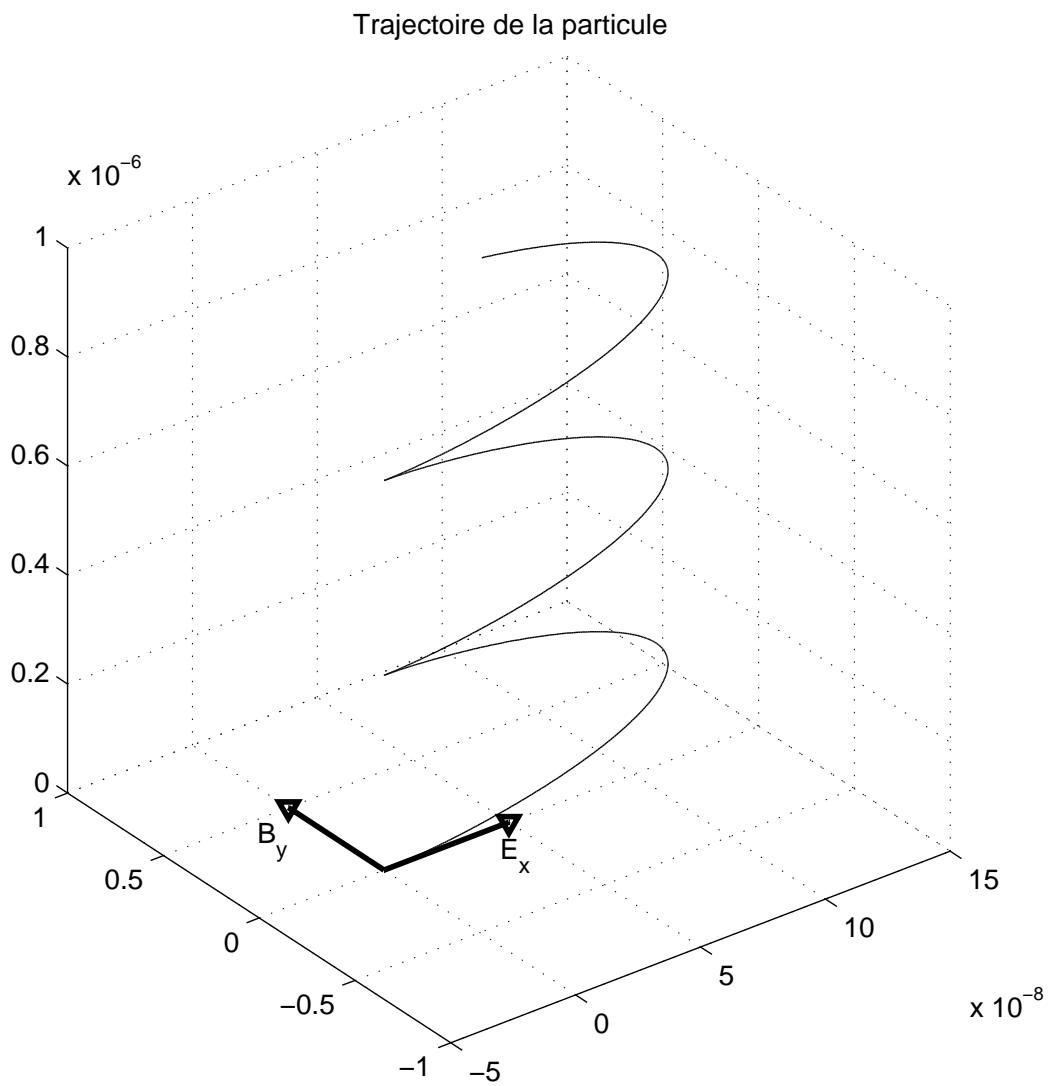


FIG. 9: Trajectoire d'une particule chargée dans un champ électrique  $\vec{E} = (E_x; 0; 0)$  et magnétique  $\vec{B} = (0; B_y; 0)$

## Références

- [1] D. M. Etter : “Engineering problem solving with Matlab”, Prentice Hall, 1993 27, 30
- [2] W. J. Palm III : “Introduction to Matlab for Engineers”, WCB McGraw-Hill, 1998
- [3] M. Etique : “Introduction au logiciel Matlab, cours bloc”, eivd, 2001 1
- [4] M. Correvon :”Introduction à Matlab 5.2 et Simulink 2.2”, Laboratoire de Systèmes Electromécaniques, eivd, 2002 1
- [5] E. Lambert : “Cours de physique”, eivd, 1999 16, 19
- [6] [www.mathworks.com](http://www.mathworks.com) : site officiel de Matlab
- [7] [www.scientific.de](http://www.scientific.de) : site européen de Matlab
- [8] [www.sysquake.com](http://www.sysquake.com) : site d’un logiciel similaire à Matlab dont la version démonstration est gratuite
- [9] [www-rocq.inria.fr/scilab/](http://www-rocq.inria.fr/scilab/) : site d’un logiciel libre de la communauté Linux et comparable à Matlab
- [10] [www.math.siu.edu/Matlab/tutorials.html](http://www.math.siu.edu/Matlab/tutorials.html) : site proposant des tutoriaux intéressants et avancés illustrant les possibilités de Matlab

## 7 Exercices

### 7.1 Résolution d'un circuit électrique

Étant donné le circuit de la figure 10, calculez les valeurs efficaces et phases des courants et la tension aux bornes de  $R_1$  et  $L$  lorsque

$$U_g = 220 [V], \quad R_1 = 100 [\Omega], \quad R_2 = 200 [\Omega], \\ L = 1 [mH], \quad C_1 = 10 [\mu F], \quad C_2 = 100 [nF]$$

et que la fréquence du générateur vaut 50Hz et 250Hz. Plutôt que de modifier la valeur de la fréquence dans le fichier, utilisez la commande `input` :

```
frequence = input ('Entrez la frequence du generateur : ');
```

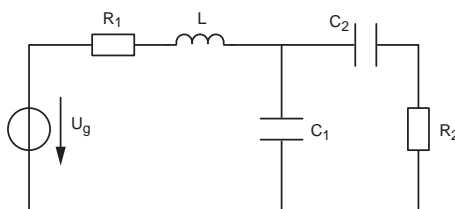


FIG. 10: Résolution d'un circuit électrique

### 7.2 Réponses fréquentielle et temporelle d'un circuit

On désire réaliser et analyser un filtre passif passe-bas d'ordre 2 :

1. Dessinez le schéma du filtre constitué d'une résistance  $R$ , d'une inductance  $L$  et d'une capacité  $C$ .
2. Calculez littéralement sa fonction de transfert.
3. Admettant  $L = 10 [mH]$ ,  $C = 10 [nF]$  et  $R = 500, 1000, 2000 [\Omega]$ , calculez et tracez les graphes similaires à ceux des figures 2 et ???. Adaptez le domaine fréquentiel à celui du circuit.
4. Faites de même pour les réponses temporelles (figure ??). Si nécessaire, modifiez le domaine temporel.

### 7.3 Réponses fréquentielle et temporelle d'un filtre

La réalisation de filtres quelconques se fait à partir d'une cellule de base biquadratique décrite par un polynôme d'ordre 2 au numérateur et au dénominateur. Suivant la valeur des coefficients du polynôme du numérateur, on obtiendra un filtre passe-bas, passe-bande, passe-haut et réjecteur de bande.

Des filtres d'ordre supérieur à 2 sont réalisés en plaçant en série des cellules d'ordre 1 ou 2. Ainsi, un filtre d'ordre 5 sera réalisé par la mise en cascade d'une cellule d'ordre 1 et de 2 cellules d'ordre 2.

Sachant que les cellules d'ordre 2 de type passe-bas sont généralement décrites comme suit :

$$H_{PB}(j\omega) = A_u \frac{1}{1 + \frac{1}{Q_0} \left(\frac{j\omega}{\omega_n}\right) + \left(\frac{j\omega}{\omega_n}\right)^2}$$

où  $A_u$  est le gain,  $Q_0$  le facteur de qualité et  $\omega_n$  la pulsation caractéristique de la cellule d'ordre 2, on demande :

1. Écrivez la fonction de transfert de chaque cellule réalisant un filtre passe-bas d'ordre 6 sachant que
  - a) les 3 cellules ont la même pulsation caractéristique  $\omega_n = 1000 [rad/sec]$  ;
  - b) les facteurs de qualité sont  $Q_{01} = 0.5176$ ,  $Q_{02} = 0.7071$  et  $Q_{03} = 1.9319$ .
2. Calculez la fonction de transfert globale en effectuant le produit des 3 fonctions de transfert ; affichez le résultat avec la commande `zpk`.
3. Tracez sur un même diagramme de Bode :
  - a) l'amplitude des réponses fréquentielles de chaque cellule en traitillé ;
  - b) la réponse fréquentielle du filtre complet en trait continu ;
  - c) une droite horizontale située au niveau  $-3dB$  en pointillé.
4. Par programmation, recherchez la pulsation de coupure de ce filtre ; que vaut-elle ? (la pulsation de coupure correspond à une amplitude diminuée d'un facteur  $\sqrt{2}$  par rapport à sa valeur maximum).
5. Tracez la réponse indicielle du filtre.

*Remarque :* Les valeurs proposées pour la réalisation du filtre correspondent à celles d'un filtre, dit de Butterworth, dont la réponse fréquentielle est plate au maximum dans la bande passante.

## 7.4 Portance d'une aile

Lors des essais de portance d'une aile en fonction de son inclinaison, on a mesuré les valeurs suivantes [1] :

Angle [deg]	-5	0	5	10	15	17	19	21
Force [N]	-2.80	1.08	5.00	8.80	11.9	12.4	12.4	11.8

1. Tracez le graphe de la portance. Mettez en valeur ce graphe à l'aide d'un titre et d'informations sur l'abscisse et l'ordonnée.
2. Recherchez une loi polynomiale représentant au mieux cette caractéristique. Au lieu de modifier l'ordre du polynôme dans le fichier lui-même, utilisez la commande `input`.
3. Sur un même diagramme, tracez les points mesurés avec le symbole 'o' et en continu, la courbe polynomiale pour un angle compris entre  $-10$  et  $+25$  degrés.
4. Sachant que la mesure de la force est entachée d'une erreur de  $\pm 0.5 [N]$ , reprenez ce diagramme et tracez les points mesurés avec les domaines d'erreur et, en continu, la courbe polynomiale.
5. Quel est le polynôme qui vous paraît le mieux adapté pour représenter cette aile ?

## 7.5 Modélisation de la caractéristique d'une diode

Une diode semi-conductrice est un élément électronique dont le fonctionnement peut être représenté par l'équation suivante :

$$I_d = I_s \left( e^{U_d/nV_T} - 1 \right)$$

avec :

$I_d$	courant traversant la diode
$U_d$	tension mesurée aux bornes de la diode
$I_s$	courant de saturation inverse
$V_T = kT/q$	potentiel thermique valant 26 mV @ T = 300 K
$n$	facteur technologique ( $1 < n < 1.8$ pour les diodes Si)

Lorsque la diode est conductrice, cette loi s'écrit plus simplement :

$$I_d = I_s e^{U_d/nV_T}$$

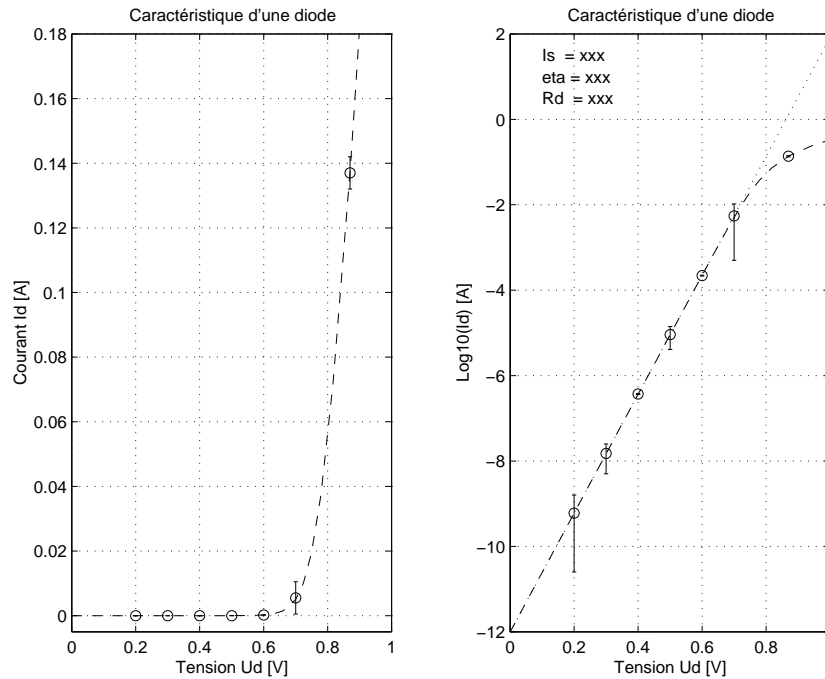


FIG. 11: Modélisation de la caractéristique d'une diode

On dit alors que le courant croît exponentiellement avec la tension appliquée à la diode ou, de manière inverse, que la tension aux bornes de la diode croît logarithmiquement avec le courant :

$$U_d = n V_T \ln \left( \frac{I_d}{I_s} \right)$$



Lorsque le courant traversant une diode réelle est élevé, on doit encore tenir compte d'une chute de tension ohmique due à la résistance de liaison entre la diode idéale et les fils de connexion. La chute de tension mesurée aux bornes d'une diode réelle est alors décrite par :

$$U_d = n V_T \ln \left( \frac{I_d}{I_s} \right) + R_d I_d$$

Sachant que l'on a mesuré sur une diode les tensions et courants présentés dans le tableau suivant :

$U_d [V]$	0.2	0.3	0.4	0.5	0.6	0.7	0.87
$I_d$	0.6 nA	15 nA	350 nA	9.0 $\mu$ A	220 $\mu$ A	5.5 mA	137 mA
$\pm \Delta I_d$	1 nA	10 nA	10 nA	5 $\mu$ A	5 $\mu$ A	5 mA	5 mA

1. Avec Matlab, calculez et dessinez les graphes de la figure 11.
2. Indiquez les domaines d'erreur avec la commande `errorbar`.
3. Ajustez successivement les paramètres  $I_s$  et  $n$  pour obtenir la droite en pointillé passant au mieux parmi les points mesurés puis variez  $R_d$  pour obtenir la courbe en traitillé.
4. Affichez sur le graphe la valeur des paramètres  $I_s$ ,  $n$  et  $R_d$  finalement trouvés.
5. Quelle(s) méthode(s) mathématique(s) pourriez-vous utiliser pour trouver automatiquement ces paramètres ?

## 7.6 Chute libre

On considère dans cet exercice un parachutiste qui saute d'un ballon à vitesse nulle et descend en chute libre. Les variables utiles à la résolution sont sa position  $y(t)$  orientée vers le bas, sa vitesse  $v(t)$  ou  $\dot{y}(t)$  et son accélération  $a(t)$  ou  $\ddot{y}(t)$ .

L'équation du mouvement du parachutiste découle de la loi de Newton :

$$m\ddot{y}(t) = m g - F_{air}(t)$$

On considère généralement que la force de frottement aérodynamique est proportionnelle au carré de la vitesse ; on a alors :

$$F_{air}(t) = +\mu \text{ signe}(v(t)) v(t)^2$$

Admettant que :

- la masse  $m$  du parachutiste est de 80 [kg],
- la vitesse maximum atteinte est d'environ 200 [km/h],

écrivez un programme Matlab répondant aux points suivants :

1. Sachant que le frottement de l'air compense exactement le poids du parachutiste lorsque la vitesse maximum est atteinte, calculez le coefficient  $\mu$  du frottement aérodynamique.
2. Résolvez l'équation différentielle de l'évolution du parachutiste et déterminez le temps et la hauteur nécessaires pour atteindre le 98% de la vitesse limite.

## 7.7 Saut à l'élastique

Dans ce qui suit, on se propose d'aborder le problème du saut à l'élastique par une modélisation progressive [1]. Les paramètres nécessaires à sa description sont :

- la masse  $m = 80 [kg]$  du sauteur,
- le coefficient  $\mu = 0.25 [N/(m/s)^2]$  du frottement aérodynamique
- la longueur  $L = 150 [m]$  de l'élastique,
- le coefficient d'élasticité  $k = 10 [N/m]$  de l'élastique,
- le coefficient de frottement interne  $\lambda = 7 [N/(m/s)]$  de l'élastique.

Les variables utiles à la résolution du problème sont :

- la position  $y(t)$ ,
- la vitesse  $v(t)$  ou  $\dot{y}(t)$ ,
- l'accélération  $a(t)$  ou  $\ddot{y}(t)$ .

L'équation du mouvement du sauteur découle de la loi de Newton :

$$m\ddot{y}(t) = mg - F_{air}(t) - F_{elast}(t)$$

avec :

$$F_{air}(t) = +\mu \text{ signe}(v(t)) v(t)^2$$
$$F_{elast}(t) = \begin{cases} k(y(t) - L) + \lambda v(t) & \text{si } y(t) > L \\ 0 & \text{si } y(t) \leq L \end{cases}$$

Écrivez un programme Matlab répondant aux points suivants :

1. Résolvez l'équation différentielle en considérant un élastique sans frottements internes ( $\lambda = 0$ ).
2. Tracez les graphes de position, vitesse et accélération ; vous paraissent-ils raisonnables ?
3. Résolvez l'équation différentielle prenant en compte les frottements de l'élastique.
4. Tracez les graphes ; quelle est l'importance des frottements internes ?
5. A quel instant et quelle vitesse, l'élastique commence-t-il à se tendre ?
6. Que valent la vitesse maximum et l'accélération maximum ? A quels instants apparaissent-elles ?

## 7.8 Particule dans un champ électromagnétique

On considère une particule de masse  $m = 50 \times 10^{-31} [kg]$  et de charge  $q = 8 \times 10^{-19} [C]$  se déplaçant dans un champ électrique uniforme  $\vec{E} = (1; 1; 1) [V/m]$  et un champ magnétique uniforme  $\vec{B} = (0.0; 0.1; 0.0) [T]$ . Sachant qu'en l'instant  $t = 0$ , la particule quitte l'origine du référentiel avec une vitesse nulle :

1. Calculez les composante de la vitesse et la position de la particule au cours du temps.
2. Tracez des graphes similaires à ceux de la figure 12.
3. Sur un nouveau graphe 3D, dessinez les vecteurs  $\vec{E}$  et  $\vec{B}$ .

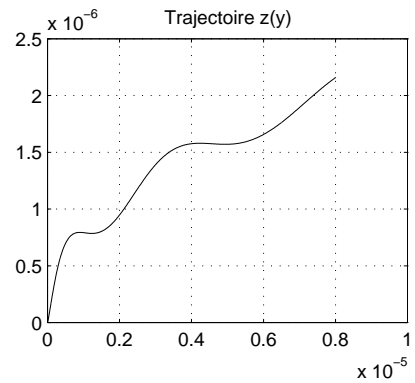
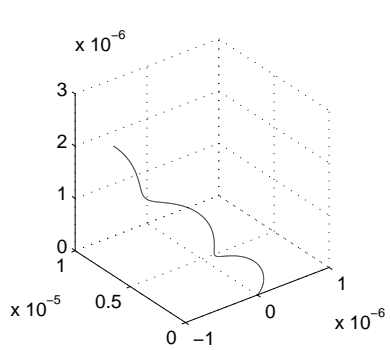
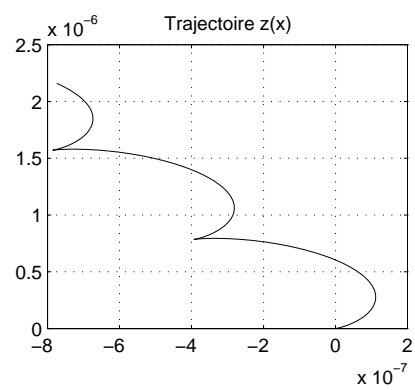
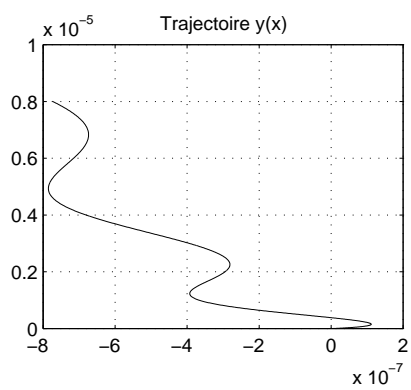


FIG. 12: Déplacement d'une particule

## 8 Liste de quelques fonctions Matlab

Cette liste, qui n'est de loin pas exhaustive, regroupe les commandes et fonctions les plus usuelles. Elle illustre également la richesse des possibilités offertes. Pour plus de détails, il faut évidemment consulter l'aide en ligne offerte par Matlab et profiter des nombreux exemples qui y sont proposés.

### 8.1 Environnement Matlab

#### 8.1.1 Commandes et fonctions

<code>help</code>	Aide
<code>what</code>	Liste les fichiers *.m présents dans le répertoire courant
<code>type</code>	Imprime un fichier
<code>lookf</code>	Recherche d'une entrée dans l'aide
<code>which</code>	Localise les fonctions et fichiers
<code>demo</code>	Lance la démonstration
<code>path</code>	Définition des chemins d'accès aux fichiers et fonctions
<code>cedit</code>	Paramètres d'édition d'une ligne de commande
<code>version</code>	Affiche le numéro de version de Matlab
<code>whatsnew</code>	Affiche les fichiers README de la toolbox
<code>info</code>	Informations sur Matlab et Mathworks
<code>why</code>	Fournit une réponse aléatoire

#### 8.1.2 Informations sur l'espace de travail

<code>who</code>	Affiche les variables courantes
<code>whos</code>	Affiche les variables courantes avec leurs dimensions
<code>save</code>	Sauve l'espace de travail sur disque
<code>load</code>	Restaure l'espace de travail à partir du disque
<code>clear</code>	Efface les variables et fonctions de la mémoire
<code>close</code>	Ferme la fenêtre courante
<code>pack</code>	Réorganise la mémoire
<code>size</code>	Renvoie la taille d'une matrice
<code>length</code>	Renvoie la longueur d'un vecteur
<code>disp</code>	Affiche une matrice de texte

#### 8.1.3 Commandes système

<code>cd</code>	Change le répertoire courant
<code>pwd</code>	Affiche le répertoire courant
<code>dir, ls</code>	Liste les fichiers
<code>delete</code>	Suppression de fichiers
<code>getenv</code>	Renvoie la variable d'environnement
<code>!</code>	Appelle et exécute une commande système
<code>diary</code>	Sauvegarde le texte d'une session Matlab

#### 8.1.4 Fenêtre de commandes

<code>clc</code>	Efface le contenu de la fenêtre de commandes
<code>home</code>	Place le curseur en haut de l'écran
<code>format</code>	Définit le format d'affichage
<code>echo</code>	Affiche les instructions exécutées par un script
<code>more</code>	Contrôle de l'affichage paginé
<code>quit, exit</code>	Ferme Matlab
<code>Matlabrc</code>	Fichier de démarrage

#### 8.1.5 Caractères spéciaux

<code>[ ]</code>	Définition de matrices ou vecteurs ; enferme les arguments de sortie des fonctions
<code>( )</code>	Gère la priorité des opérations ; enferme les arguments d'entrée des fonctions
<code>.</code>	Point décimal
<code>..</code>	répertoire parent
<code>...</code>	Indique une ligne suite
<code>,</code>	Séparateur d'arguments ou d'instructions
<code>;</code>	Fin de lignes (matrices) ou suppression de l'affichage
<code>%</code>	Commentaires
<code>:</code>	Manipulation de sous matrices ou génération de vecteurs
<code>!</code>	Appel système

#### 8.1.6 Opérateurs logiques

<code>&lt;</code>	Inférieur à
<code>&amp;</code>	Et
<code>&gt;</code>	Supérieur à
<code> </code>	Ou
<code>&lt;=</code>	Inférieur ou égal à
<code>~</code>	Non
<code>&gt;=</code>	Supérieur ou égal à
<code>xor</code>	Ou exclusif
<code>==</code>	Egal à
<code>~=</code>	Différent de
<code>=</code>	Assignation

### 8.1.7 Variables prédéfinies, durée, date

<code>ans</code>	Réponse à une expression sans assignation
<code>eps</code>	Précision de la virgule flottante
<code>realmax</code>	Plus grand nombre flottant
<code>realmin</code>	Plus petit nombre flottant positif
<code>pi</code>	$\pi$
<code>i, j</code>	$\sqrt{-1}$
<code>inf</code>	$\infty$
<code>NaN</code>	Not a Number
<code>flops</code>	Nombre d'opérations flottantes par seconde
<code>nargin</code>	Nombre d'arguments d'entrée d'une fonction
<code>nargout</code>	Nombre d'arguments de sortie d'une fonction
<code>computer</code>	Type du calculateur
<code>date</code>	Date courante
<code>clock</code>	Horloge
<code>etime</code>	Durée d'exécution
<code>tic, toc</code>	Affiche le début et la fin d'exécution
<code>cputime</code>	Temps CPU écoulé

### 8.1.8 Fonctions logiques

<code>exist</code>	Teste l'existence d'une variable ou d'une fonction
<code>any</code>	Vrai si un élément est vrai
<code>all</code>	Vrai si tous les éléments sont vrais
<code>find</code>	Cherche l'indice des éléments non nuls
<code>isnan</code>	Vrai si l'élément n'est pas un nombre
<code>isinf</code>	Vrai pour tout élément infini
<code>finite</code>	Vrai pour tout élément fini
<code>isieee</code>	Vrai si la représentation est au format IEEE
<code>isempty</code>	Vrai pour une matrice vide
<code>issparse</code>	Vrai pour une matrice creuse
<code>isstr</code>	Vrai pour une chaîne de caractères
<code>strcmp</code>	Comparaison de deux chaînes

### 8.1.9 Instructions de contrôle

<code>if, else, elseif</code>	Test conditionnel
<code>for</code>	Instruction de répétition avec compteur
<code>while</code>	Instruction de répétition avec test
<code>end</code>	Terminaison de <code>if</code> , <code>for</code> et <code>while</code>
<code>break</code>	Interrompt une boucle
<code>return</code>	Retour
<code>error</code>	Affiche un message et interrompt l'exécution

### 8.1.10 Instructions spécifiques

<code>input</code>	Indicateur d'attente d'entrée
<code>keyboard</code>	Considère le clavier comme un fichier script
<code>menu</code>	Génère un menu de choix pour l'utilisateur
<code>pause</code>	Attente
<code>function</code>	Définition de fonction
<code>eval</code>	Exécute une chaîne de caractère
<code>feval</code>	Exécute une fonction définie dans une chaîne
<code>global</code>	Définit les variables comme globales
<code>nargchk</code>	Valide le nombre d'arguments d'entrée

## 8.2 Fonctions mathématiques

### 8.2.1 Fonctions élémentaires

<code>abs</code>	Valeur absolue ou module d'une grandeur complexe
<code>angle</code>	Argument d'une grandeur complexe
<code>sqrt</code>	Racine carrée
<code>real</code>	Partie réelle
<code>imag</code>	Partie imaginaire
<code>conj</code>	Complexe conjugué
<code>gcd</code>	PGCD
<code>lcm</code>	PPCM
<code>round</code>	Arrondi à l'entier le plus proche
<code>fix</code>	Troncature
<code>floor</code>	Arrondi vers le bas
<code>ceil</code>	Arrondi vers le haut
<code>sign</code>	Signe de
<code>rem</code>	Reste de la division
<code>exp</code>	Exponentielle
<code>log</code>	Log népérien
<code>log10</code>	Log décimal
<code>log2</code>	Log base 2
<code>pow2</code>	Calcule 2 à la puissance y

### 8.2.2 Fonctions trigonométriques

`sin, asin, sinh, asinh`  
`cos, acos, cosh, acosh`  
`tan, atan, tanh, atanh`  
`cot, acot, coth, acoth`  
`sec, asec, sech, asech`  
`csc, acsc, csch, acsch`

### 8.2.3 Fonctions prédéfinies

bessel	Fonction de Bessel
beta	Fonction beta
gamma	Fonction gamma
rat	Approximation par un rationnel
rats	Format de sortie pour rat
erf	Fonction erreur erf
erfinv	Inverse de erf
ellipke	Intégrale elliptique complète
ellipj	Fonction elliptique de Jacobi
expint	Fonction intégrale exponentielle pour n=1

## 8.3 Matrices et algèbre linéaire

### 8.3.1 Opérateurs sur les matrices

+, -, *, ^	Addition, Soustraction, Multiplication, Puissance
/, \	Division à droite, Division à gauche
'	Transposition conjuguée

### 8.3.2 Opérateurs sur les composantes matricielles

+, -, .*, . ^	Addition, Soustraction, Multiplication, Puissance
./, .\	Division à droite, Division à gauche
.'	Transposition

### 8.3.3 Manipulation des matrices

diag	Création ou extraction de la diagonale
rot90	Rotation de 90 degrés
fliplr	Retournement gauche-droit
flipud	Retournement haut-bas
reshape	Redimensionnement
tril	Partie triangulaire inférieure
triu	Partie triangulaire supérieure
.'	Transposition
:	Conversion matrice -> vecteur



### 8.3.4 Matrices prédéfinies

zeros	Matrice de 0
ones	Matrice de 1
eye	Matrice identité
diag	Matrice diagonale
toeplitz	Matrice de Toeplitz
magic	Carré magique
compan	Matrice compagnon
linspace	Vecteurs à composantes linéairement espacées
logspace	Vecteurs à composantes logarithmiquement espacées
meshgrid	Grille pour les graphiques 3D
rand	Nombres aléatoires à répartition uniforme
randn	Nombres aléatoires à répartition normale
hilb	Hilbert
invhilb	Inverse de Hilbert (exact)
vander	Vandermonde
pascal	Pascal
hadamard	Hadamard

### 8.3.5 Opérations sur les matrices

poly	Polynôme caractéristique
det	Déterminant
eig	Valeurs propres
trace	Trace
inv	Inversion

### 8.3.6 Décomposition et factorisation de matrices

lu	Décomposition LU
rref	Réduction de lignes
chol	Factorisation de Cholesky
qr	Décomposition QR
nuls	Moindres carrés non-négatif
ris	Moindres carrés avec covariance connue
null	Noyau
orth	Orthogonalisation
eig	Valeurs et vecteurs propres
hess	Forme de Hessenberg
schur	Décomposition de Schur,
cdf2rdf	Forme complexe diagonale vers forme réelle diagonale par blocs
rsf2csf	Forme réelle diagonale par blocs vers forme complexe diagonale
balance	Mise à l'échelle pour le calcul des valeurs propres
qz	Valeurs propres généralisées
polyeig	Polynôme aux valeurs propres
svd	Décomposition en valeurs singulières
pinv	Pseudo-inverse

## 8.4 Textes et chaînes de caractères

<code>abs</code>	Convertit une chaîne en valeurs numériques Ascii
<code>blanks</code>	Crée une chaîne d'espaces
<code>eval</code>	Convertit un texte en code Matlab
<code>num2str</code>	Convertit un nombre en chaîne
<code>int2str</code>	Convertit un nombre entier en chaîne
<code>str2num</code>	Convertit une chaîne en nombre
<code>isstr</code>	Vrai si l'élément est une chaîne
<code>strcmp</code>	Comparaison de chaînes
<code>upper</code>	Conversion en majuscule
<code>lower</code>	Conversion en minuscule
<code>hex2num</code>	Convertit une chaîne hexadécimale en flottant
<code>hex2dec</code>	Convertit une chaîne hexadécimale en entier
<code>dec2hex</code>	Convertit un entier en une chaîne hexadécimale

## 8.5 Fonctions graphiques

### 8.5.1 Graphiques 2D

<code>plot</code>	Dessine le graphe d'une fonction
<code>loglog</code>	Graphe en échelle log-log
<code>semilogx</code>	Graphe en échelle semi-log (abscisse)
<code>semilogy</code>	Graphe en échelle semi-log (ordonnée)
<code>fill</code>	Graphe de polynômes 2D remplis
<code>polar</code>	Graphe en coordonnées polaires
<code>bar</code>	Graphe en barres
<code>stairs</code>	Graphe en marches d'escalier
<code>stem</code>	Graphe de raies
<code>errorbar</code>	Graphe avec barres d'erreur
<code>hist</code>	Histogramme
<code>rose</code>	Histogramme en coordonnées polaires
<code>compass</code>	Représentation de vecteurs à partir de l'origine
<code>feather</code>	Représentation de vecteurs sur un axe linéaire

### 8.5.2 Annotation de graphiques

<code>title</code>	Titre du graphique
<code>xlabel</code>	Légende pour l'abscisse
<code>ylabel</code>	Légende pour l'ordonnée
<code>zlabel</code>	Légende pour la cote
<code>grid</code>	Dessin d'une grille
<code>text</code>	Texte
<code>gtext</code>	Placement de texte avec la souris
<code>ginput</code>	Entrée graphique par la souris

### 8.5.3 Contrôle des fenêtres graphiques

<code>figure</code>	Ouvre une fenêtre graphique
<code>gcf</code>	Retourne le numéro de la figure courante
<code>clf</code>	Efface la figure courante
<code>close</code>	Ferme la figure courante
<code>hold</code>	Gère la surimpression
<code>ishold</code>	Etat de la surimpression
<code>subplot</code>	Sous fenêtres graphiques
<code>axes</code>	Axes en position arbitraire
<code>gca</code>	Retourne le numéro des axes courants
<code>axis</code>	Contrôle l'apparence et l'échelle des axes
<code>caxis</code>	Contrôle l'échelle des axes et de la pseudocouleur
<code>whitebg</code>	Dessine sur fond blanc
<code>cinvert</code>	Vidéo inverse

### 8.5.4 Sauvegarde et copie graphique

<code>print</code>	Imprime ou sauve un fichier
<code>printopt</code>	Configuration de l'imprimante
<code>orient</code>	Orientation paysage ou portrait

### 8.5.5 Objets 3D

<code>sphere</code>	Génération de sphères
<code>cylinder</code>	Génération de cylindres
<code>peaks</code>	Démonstration

### 8.5.6 Animations

<code>moviein</code>	Initialise l'espace mémoire pour l'animation
<code>getframe</code>	Enregistre une image pour l'animation
<code>movie</code>	Joue l'animation

### 8.5.7 Apparence des graphiques

<code>view</code>	Spécifie l'angle de vue
<code>viewmtx</code>	Matrice de transformation
<code>hidden</code>	Gère les lignes cachées
<code>shading</code>	Mode de remplissage
<code>specular</code>	Réfectance d'une surface
<code>diffuse</code>	Réfectance d'une surface
<code>surfnorm</code>	Calcule la surface normale
<code>colormap</code>	Table de correspondances couleurs
<code>brighten</code>	Surbrillance ou sous brillance pour colormap
<code>spinmap</code>	Change colormap de manière cyclique
<code>rgbplot</code>	Dessine la colormap
<code>hsv2rgb</code>	Conversion hsv vers rgb
<code>rgb2hsv</code>	Conversion rgb vers hsv

### 8.5.8 Graphiques tridimensionnels

<code>mesh</code>	Surface maillée
<code>meshc</code>	Combinaison <code>mesh</code> + dessin des équi-niveaux
<code>meshz</code>	Surface maillée avec plan de référence
<code>surf</code>	Surface 3D à facettes
<code>surf c</code>	Combinaison <code>surf</code> + dessin des équi-niveaux
<code>surf1</code>	Surface 3D à facettes avec éclairage
<code>plot3</code>	Dessin de lignes et points en 3D
<code>plot3</code>	<code>fill3</code> Graphe de polynômes 3D remplis
<code>contour</code>	Dessin 2D des équi-niveaux
<code>contourc</code>	Calcul des valeurs utilisées par <code>contour</code>
<code>contour3</code>	Dessin 3D des équi-niveaux
<code>clabel</code>	Étiquettes des équi-niveaux (contours)
<code>pcolor</code>	Dessine en pseudocouleur
<code>quiver</code>	Affichage du gradient sous forme de flèches
<code>image</code>	Affiche une image
<code>waterfall</code>	Représentation chute d'eau
<code>slice</code>	Visualisation en volume

### 8.5.9 Opérations sur les objets graphiques

<code>uicontrol</code>	Création d'un interface de contrôle utilisateur
<code>uimenu</code>	Création d'un interface menu utilisateur
<code>set</code>	Définit les propriétés d'un objet
<code>get</code>	Lit les propriétés d'un objet
<code>reset</code>	Réinitialise les propriétés d'un objet
<code>delete</code>	Supprime un objet
<code>drawno</code>	Force les évènements graphiques en attente

### 8.6 Opérations sur les polynômes

<code>poly</code>	Construit un polynôme à partir des racines
<code>roots</code>	Calcul des racines
<code>polyval</code>	Valeur d'un polynôme en un point
<code>polyvalm</code>	Valeurs d'un polynôme en une matrice de points
<code>conv</code>	Multiplication de deux polynômes
<code>deconv</code>	Division d'un polynôme par un autre
<code>residue</code>	Décomposition en éléments simples et résidus
<code>polyfit</code>	Polynôme d'approximation
<code>polyder</code>	Différentiation

## 8.7 Analyse de données et statistiques

### 8.7.1 Analyse de données par colonne

max	Valeur max
min	Valeur min
mean	Valeur moyenne
median	Valeur médiane
std	Ecart type
sort	Tri en ordre croissant
sum	Somme des éléments
prod	Produit des éléments
cumsum	Vecteur des sommes partielles cumulées
cumprod	Vecteur des produits partiels cumulés
hist	Histogramme

### 8.7.2 Analyse et traitement des signaux

corrcoef	Coefficients de corrélation
cov	Matrice de covariance
filter	Filtrage monodimensionnel
filter2	Filtrage bidimensionnel
cplxpair	Tri en paires complexes
unwrap	Suppression des sauts de phase
nextpow2	Puissance de 2 immédiatement supérieure
fft	FFT monodimensionnelle (fréquences de 0 à 1)
ffr2	FFT bidimensionnelle
ift	FFT inverse
ifft2	FFT inverse
fftshift	FFT (fréquences de -1/2 à 1/2)

## 8.8 Intégration, interpolation et et dérivation numériques

### 8.8.1 Intégration numérique

quad	Intégrale de Simpson
quad8	Intégrale de Newton-Cotes
trapz	Méthode des trapèzes

### 8.8.2 Interpolation

spline	Interpolation spline cubique
interp1	Interpolation monodimensionnelle
interp2	Interpolation bidimensionnelle
interpft	Interpolation 1D par FFT
griddata	Maillage de données

### 8.8.3 Différences finies

diff	Approximation de la dérivée
gradient	Approximation du gradient
del2	Laplacien sur 5-points

### 8.9 Optimisation et équations non linéaires

fmin	Minimisation d'une fonction d'une variable
fmins	Minimisation d'une fonction de plusieurs variables
fsolve	Résolution d'un système d'équations non-linéaires
fzero	Zéro d'une fonction d'une variable

### 8.10 Modélisation et analyse de systèmes continus

Les fonctions qui suivent, malgré le fait qu'elles sont très fréquemment utilisées dans divers domaines, n'appartiennent malheureusement pas aux fonctions de base. Pour pouvoir les utiliser, il faut acquérir la Control Toolbox !

tf	Création d'une fonction de transfert
tfddata	Extraction du numérateur et du dénominateur d'une fonction de transfert
minreal	Suppression des pôles compensés par des zéros (simplification algébrique)
damp	Fréquence propre et amortissement/résonnance
dcgain	Gain en continu
pzmap	Position des pôles et zéros dans le plan complexe
roots	Racines d'un polynôme

#### 8.10.1 Construction d'un modèle

ord2	Création d'un modèle continu d'ordre 2
feedback	Mise en réaction (positive ou négative) d'un système
parallel	Connexion parallèle de deux modèles
series	Connexion série de deux modèles

#### 8.10.2 Réponse temporelle

step	Réponse indicielle
impulse	Réponse impulsionnelle
initial	Condition initiale pour une réponse temporelle
ltiview	Visionneuse pour l'analyse de la réponse de systèmes linéaires

#### 8.10.3 Réponse fréquentielle

bode	Diagramme de Bode
freqresp	Réponse à une gamme de fréquence
linspace	Vecteurs linéairement espacés
logspace	Vecteurs logarithmiquement espacés
nichols	Diagramme de Nichols
ngrids	Grille pour les diagrammes de Nichols
nyquist	Diagramme de Nyquist