Outils mathématiques et numériques

Riccardo Spezia

Chercheur CNRS au laboratoire LAMBE

LAMBE Laboratoire Analyse et Modélisation pour la Biologie et l'Environnement, UMR-CNRS 8587, Université d'Evry val d'Essonne, Bât. Maupertuis, F-91025 Evry - France

riccardo.spezia@univ-evry.fr
http ://www.lambe.univ-evry.fr/rspezia

Formation prédoctorale en chimie 2015-2016
Première année (L3 – S1)

Table des matières

Chapitre 1

Introduction

Chapitre 2

Cours

2.1 Cours 1 – 06/01. Introduction

Donner des instructions de base à un ordinateur. Les fichiers sources. Editer un ASCII : vi, emacs, nedit , etc ... Languages (fortran77, fortran90, fortran95, c, c++ ...) : compilation et obtention d'un executable. Scripts (interpretes : perl, awk, python ...) : execution directe.

Pour l'ordinateur les variables peuvent être de types différentes (concept général , ici exemple en fortran90). Entier (Integer) : 4 byte ; Réels (réal), peuvent être à précision single (kind=4 ; 6 chiffres significative) ou double (kind=8 ; 15 chiffres significatives) ; Charactèr (character) : il faut préciser le nombre de char (len=n). Chaque caractère est représenté par son code ASCII et occupe 1byte ; Logique (logical) : True ou False ; Complex (deux nombres pour chaque variable).

2.1.1 Graphisme, MATLAB comme environnement

Graphisme, MATLAB comme environnement (voir exemples de algorithmes, fonctions etc ....). I/O.

D'abord les commands du "prompt". On peut utiliser le prompt directement, en tapant une instruction et ensuite la touche Entrée.

Ex 1. Taper dans le prompt :

» -1.3*sqrt(958)+cos(3*pi/4)

Matlab répond :

» ans = -40.9442

On peut aussi donner une nom à l'expression.

Ex 2. Taper dans le prompt :

» result = -1.3*sqrt(958)+cos(3*pi/4)

Matlab répond :

» result = -40.9442

On peut trouver beaucoup de matériel en ligne :

www.mathworks.fr/moler/

www.mathworks.fr/academia/teaching_kits/

Commandes d'édition :

↑ : rappelle la commande précédente

↓ : rappelle la commande suivante

w ↑ : rappelle la dernière commande commençant par w

← : déplace le curseur d'un caractère à gauche

→ : déplace le curseur d'un caractère à droite

ctrl ← : déplace le curseur d'un mot à gauche

ctrl → : déplace le curseur d'un mot à droite

clc : efface l'écran

Faire quelque autre commande de base (avec et sans " ;').

Voir commands : format , clear, help, doc

Priorité dans les opérations :

exponentielle – division (mulitplication) – addiction (substraction)

e.g. : 3+5-2^3 et 3+(5-2)^3

Il est conseillée d'utiliser toujours les parenthèses.

Constants permanentes :

pi, realmax, realmin, eps(le nombre plus proche à 1)

e.g. : x=exp(192^2)

Nombres complexesi : 2i, 3+4i , 4*exp(i*pi/4) compass(z) pour les visualiser (voire help compass et doc compass).

Vecteurs

Création de vecteurs

```
x = [3 4 5 10]
x = linspace(0,10,11)
x = 0 :10
p = 0 :2 :10
```

Commandes pour les vecteurs :

```
size(x) : taille
length(x) : taille
```

Example

```
x=1 :5
y=sin(x)
x(3)=[] (supprime l'élément 3)
x=ones(1,4)
x=zeros(1,4)
t(4)=5 (pour ajouter un élément)
```

Vecteurs colonne

```
x=(1 :1 :10)'
x=[1 4 9]'
x=[1 ;4 ;9]
x=ones(4,1)
x=zeros(4,1)
```

Example : la sum de première n nombres (exercice de Gauss) : n(n+1)/2

x=1 :1 :100

sum(x)

Autres commandes sur les vecteurs

cumsum(x)

prod(x)

prod(1 :10) (factoriel)

sort(x)

min(x)

max(x)

diff(x)

norm(x) ($||x|| = \sqrt{\sum_i x_i^2}$)

floor(x)

sign(x)

mean(x)

std(x)

Operations composante par composante

.* produit

./ division

.^ exponentiel

Produit scalaire :

u*v' o v*u' o dot(u,v)

Pour s'amuser :

funtool

```
load handel
sound(y)

A=randn(5000,2);
sound(A)
```

Plot

Pour visualiser deux vecteurs x,y de la même longuer

plot(x,y,'options')

options couleur : y (jaune), m (magenta), c (chan), r (rouge), g (vert), b (bleu), w (blanc), k (noir)
options style : -, o, x, *, +, : , - , -. , –, > , <, p (pentagramme) etc ...

Exemple :

```
x=-5 ;0.1 :5 ;
plot(x,sin(x),'r') ;
axis([-6 6 -2 2])
```

Pour changer l'échelle de l'axe nous avons aussi les commandes :

```
axis equal
axis('auto')
axis square
```

Pour plotter sur la même figure

```
figure
hold on
plot(x,f1(x),options) ;
plot(x,f2(x),options) ;
... etc ...
hold off
```

On peut aussi plotter directement sur la même figure : plot(x,sin(x),x,exp(-x),x,x.^2) ;

Avec subplot(n,m,i) nous pouvons construire une matrice (n,m) où placer les graphes

```
subplot(2,2,1)
plot(x)
subplot(2,2,2)
plot(sin(x))
subplot(2,2,3)
plot(cos(x))
subplot(2,2,4)
plot(exp(x))
```

Pour tracer une ligne brisée :
plot([1 -1 3],[7 5 -2]) ;
plot([1 -1 3 1],[7 5 -2 7]) ;

Pour tracer une fonction (voir après pour fonction décrite dans un fichier) : plot(x,fonc(x))

Nous pouvons décrire une fonction par la commande inline. Exemple :

```
x=-5 :0.2 :5 ; alpha=7 ;
y = inline('1./(alpha+x.^2)')
plot(x,y(alpha,x))
```

Faire des exemples en traçant la fonction en changeant un paramètre (figure, hold on ... hold off)

On peut tracer une fonction avec la commande fplot(donc, [xmin xmax y min ymax], options)
E.g. : fplot('sin',[0 pi])

Pour les outils symbolic nous pouvons utiliser ezplot('fonc(x)',[xmin xmax])
E.g. : ezplot('x^2+y^2-4')

Nous pouvons écrire sur un graphe directement de ligne de commande (mais aussi par interface graphique) xlabel('temp')
ylabel('temp')

title('bella')

title('tan\beta')

gtext('square')

legend('courbe1','courbe2',...)

Nous pouvons changer les graphes en cliquant.

Nous pouvons construire et dessiner des courbes 3D (surfaces) et des courbes de niveau.

E.g.

```
x=0 :0.1 :5 ;
y=-2 :0.1 :3 ;
[X,Y] = meshgrid(x,y) ;
z=sin(X).*cos(Y) ;
mesh(X,Y,z)
figure
contour(X,Y,z)
```

Pour dessiner la surface et la courbe de niveau directement :

```
meshc(X,Y,z)
```

Pour afficher la valeur des niveaux :

```
c=contour(X,Y,z) ;
clabel(c)
```

Et pour le choisir à l'avance v=[0.1 0.3 0.5 0.9] ;

```
c=contour(X,Y,z,v)
clabel(c,v)
```

2.1.2 Matrices, algèbre matricielle, scripts, instructions

TD : **Exemple** Calculer la constant d'Euler définie par

$$c = \lim_{n \to +\infty} \left(\sum_{k=1}^{k=n} \frac{1}{k} - \ln(n) \right) \tag{2.1}$$

Voir euler.m comme exemple de fichier.

Matrices

A = [1 2 3 ; 4 5 6 ; 7 8 9 ; 10 11 12]
size(A)
length(A)
sum(A)
diag(A)

M=magic(N) : matrice NxN

TD : définir des matrices magic de taille 3, 4 et 5 et calculer la somme des élément de chaque colonne, ligne, et de la diagonale.

somme de chaque colonne : sum(M)
somme de chaque ligne : sum(M')'
somme de la diagonale : sum(diag(M))

Commandes sur une matrice

A(i,j) : désigne l'élément de la ligne i et de la colonne j
b=A(3 : 4 , 2 : 3) : sélectionne dans les lignes 3 à 4 et des colonnes 2 à 3
b=A(:,2) : sélectionne la deuxième colonne de A
b=A(:,[1 3]) : sélectionne la première et la troisième colonne
b=A(:,n :-1 :-1) : renverse l'ordre des colonnes (ici, A possède n colonnes)
b=A(:) : crée un vecteur colonne constitué des colonnes de A prises dans l'ordre, en commençant par la première
b=A(:, [1 3]) = [] : supprime les colonnes 1 et 3 de A
A(4, :) = 7 : affecte la valeur 7 à tous les éléments de la ligne 4
A(4,2)=0 : modifie un élément
A=[A ; 13 14 15] ajoute une ligne

" :" signifie toute les lignes ou colonnes

Matrices spéciales :

M=eye(size(A)) , M=eye et M=eye(10)

A=zeros(5,5)

A=zeros(5)

A=ones(5)

A=rand(5,5) rand : nombres aléatoires dans l'intervalle [0,1]

A=randn(5,5) randn : nombres aléatoires en distribution gaussienne

A' est la transposée de A (si réel) et l'adjointe si complexe.

Si on clique sur A dans workspace on peut éditer directement la matrice (montrer).

Inputs Inputs de donnes et script .m

```
% calculer l'air d'un triangle
B = input('base = ');
H = input('hauteur = ');
Area = B*H/2.


% déssiner fonction sinus en choisissant la couleur
x=-pi :0.1 :pi;
color = ['r','b','g'];
choice = menu('Choose a color','Red','Blue','Green');
plot(x,sin(x),color(choice))
```

Examples de inputs : triangle.m ; color_choice.m

Algèbre matricielle

Pour les matrices carrées :

det(A) : déterminant de A

inv(A) : inverse de A

eig(A) : vecteurs des valeurs propres de A

[V,D] = eig(A) : V sont les vecteurs propres de A normalisé (i.e. norme euclidienne = 1) et D une matrice diagonale contenante les valeurs propres de A

poly(A) = det(λI - A) coefficients du polynome caractéristique

Pour des matrices quelconques :

rank(A) : le rang de A
trace(A) : la trace de A
null(A) : une base orthonormée du noyau de A.

Il est possible aussi de résoudre directement des systèmes linéaires. Si A est une matrice (n,n) et B un vecteur colonne (n,1) alors X = A \B résoudre le système par la méthode de Gauss. De même si B est un vecteur ligne (1,n) , X = A / B résoudre par la méthode de Gauss.

TD :

Balancer la réaction : m Cr + n O_2 -> k Cr_2O_3

Hint :

% Cr : 1m + 0n = 2k

% O : 0m + 2n = 3k

% Cr : a_{11} + a_{12} = b_1

% O : a_{21} + a_{22} = b_2

Cela est dans le fichier Cr2O3.m (exemple de script).

```
A = [1 0 ; 0 2 ] ;
B = [2 ; 3] ;
C = det(A) ;
X = A \B ;
Y = X*C ;
disp('Balanced reaction')
disp([num2str(Y(1)) 'Cr + ' num2str(Y(2)) 'O2 = ' num2str(C) 'Cr2O3'])
```

Polynomes

Les polynômes sont représentés par des vecteurs lignes dont les composantes sont les coefficients pris dans l'ordre des puissances décroissantes.

E.g. : $2x^5 - 9x^4 + 18x^3 - 19x^2 + 10x - 2$ est représenté par p = [2 -9 18 -19 10 -2]

Pour calculer la valeur d'un polynôme en un point s ont utilise la commande : polyval(p,s). E.g. : polyval(p, 1.5)

Operations sur le polynômes :

polyval(p,s) : calcule la valeur d'un polynôme en un point s

r = roots(p) : pour obtanir les racines de p (r est un vecteur colonne)

p1 = poly(r) : pour reconstituer le polynôme à partir des racines (p1 est un vecteur ligne)

solve('p(x)','x) : donne les solutions exactes, s'il en existe.

p2 = polyder(p) : donne les coefficients de la dérivée de p

p3 = polyint(p2) : primitive avec constante d'intégration nulle

conv(p,q) : produit de deux polynômes

[Q,R] = deconv(p,q) : division suivant les puissances décroissantes, Q étant le quotient et R le reste

Instructions de controle

Opérateurs relationnels : $<$, $>$, $<=$, $>=$, $==$ (égal), $\sim=$ (différent)

Opérateurs logiques : & (and) , | (or) , $\sim$ (non) , xor (or exclusif)

Une variable logique vaut 1 si elle est vraie et 0 si elle est fausse.

E.g. : m = 4 > 3 ; x = [-14 3 1 0 -1 2 -3] ; xx=(x>0)

Boucle for :

```
for variable = val_min :pas :val_max
instructions
end
```

Example : remplire une matrice d'Hilbert, ou chaque élément est

$H_{ij} = \frac{1}{i+j-1}$

voire exemple : hilbert_matrix.m

Boucle while :

```
while expression_logique
instructions
end
```

Tant que expression_logique est vraie tous les instructions sont exécutées. Exemple : trouver le premier nombre inter n tel quel le factoriel de n est un nombre a 100-digits

Voir : fact_100digits.m

Structure if :

```
if expression_logique
instructions1
else
instructions2
end
```

Exemple if et elseif dans boucle for : matr.m

Structure switch Permet de faire de sélection multiple sans elseif (il existe en fortran aussi ...)

```
switch variable
case vrai1
instructions1
case vrai2
instructions2
otherwise
instruction3
end
```

2.1.3 Fonctions

Fichier fonction Un fonction peut être lu à partir d'un fichier. Les variables passées en entré et sortie peuvent être retournées au programme (au à la shell) appellent. Les variables définies et utilisées sont locales.

La première ligne autre qu'un commentaire d'un fichier fonction est :
fonction [u, v, ...] = file_name(x,y,t, ...)

où file_name est obligatoirement le nom du fichier, x,y,t doivent être spécifiés à l'appel du fichier et u, v, ... sont les valeurs retournées. Si il s'agit d'une seul variable les crochets ne sont pas nécessaires.

Example :

```
function y = DvdSinExp(x)
y=exp(-x^2)*sin(5.*x);
```

Et faire :

```
fplot(@DvdSinExp,[-3, 3])
z=DvdSinExp(0.9)
```

Refaire la calcul de la constant d'Euler mais avec un fonction (c_euler2.m). Dans la ligne de commande : val = c_euler2(1000) donnera la même valeur de euler (mais il est une fonction, donc on peut plus facilement en changer la valeur).

```
% script euler.m
n=5000;
l=1 :n;
c=sum(1./l)-log(n)
% fonction c_euler2.m
function c = c_euler2(n)
l=1 :n;
c=sum(1./l)-log(n);
```

Excercise : Integrals Calculer : $I = \frac{2}{\sqrt{\pi}} \int_0^1 e^{-x^2} dx$ et $J = \int_1^2 \frac{sin(x)}{x} dx$ par la méthode des trapèzes. Rappelle :

$$\int_a^b f(x)dx \cong h\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{i=n-1} f(a+ih)\right] \tag{2.2}$$

$$h = \frac{b-a}{n} \tag{2.3}$$

Hint : la commande **feval** évalue la valeur d'une fonction en un ou plusieurs poins : feval('donc',a)

2.1.4 Dynamique

Pour résoudre numériquement les équations du mouvement de Newton (e.g. pour un système moléculaire) ils existent plusieurs méthodes numérique. Une méthode robuste

et simple est la méthode Velocity-Verlet. A chaque pas on obtiens les nouvelles positions et vitesses :

$$x(t+\Delta t) = x(t) + v(t)\Delta t + 0.5\frac{f(t)}{m}\Delta t^2 \tag{2.4}$$

$$v(t+\Delta t) = v(t) + 0.5\frac{f(t+\Delta t)+f(t)}{m}\Delta t \tag{2.5}$$

où Δt est le time step qu'il faut choisir pour assurer la conservation de l'énergie, m est la masse et $f(t)$ est la force exercé sur la particule qu'on obtien à partir de l'énergie potentielle, $f = -\frac{\partial V}{\partial x}$. Il faut évidemment avoir des conditions initiales.

Voire exemple verlet1.m : fonction comet pour faire bouger un point.

Exercise : sur un potentiel polynomial arbitraire d'ordre k :

$$V = \sum_{n}^{k} c_n x^n \tag{2.6}$$

suivre l'évolution de x avec choix de paramètres et conditions initiales. Plotter en suite : $H(t)$; $E_{pot}(t)$; $x(t)$; (x, v). (voir verlet_pol.m)

TD : do the same verlet but using external functions for verlet, energy and force (see verlet2.m)

2.2 Cours 2 – 13/01. Equations différentielles.

2.2.1 Résolution d'équations différentielles du 1ere ordre avec conditions initiales

Il s'agit de résoudre numériquement l'équation différentielle :

$$\begin{cases} \dot{y} = f(t, y) \\ y(t_0) = y_0 \end{cases} \tag{2.7}$$

Les équations sont résolu par des méthodes numériques différents (e.g. Runge-Kutta) à l'aide des solveurs de la classe "ode" : ode23, ode45, ode113, ode15s, ode23s.

La méthode de "base" est Runge-Kutta. Etant donné le problème avec conditions initiales et un pas h, alors :

$$\begin{cases} y_{n+1} = y_n + \frac{1}{2}h(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} = t_n + h \end{cases} \tag{2.8}$$

et les paramètres :

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_1) \\ k_3 = f(t_n + \frac{1}{2}h, y_n + \frac{h}{2}k_2) \\ k_4 = f(t_n + h, y_n + hk_3) \end{cases} \tag{2.9}$$

L'utilisation en est : [t,y] = solveur('funcdif', [t0 tf],y0,options) , ou solveur est remplacé par le nom de la méthode choisie et foncdif est le nom du fichier foncdif.m qui décrit la fonction f. La précision par défaut est 10^{-6} et elle peut être ameiloré par le choix des options. On peut remplacer 'fonc' par fonc .

Example pour résoudre $\dot{y}(t) = t - y^2(t)$:

```
function yp = eqd(t,y)
yp = t-y^2;
```

voir fichier eqd.m. Dans la fenêtre de commande en prenant t0=0, tf=5 et y0=1 :

```
[t,y]=ode45('eqd',[0  5],1);
plot(t,y)
```

Exemple Romeo et Juliet Voir love_eq1 et love_eq2 avec les fonctions rj1 et rj2

R(t) : Romeo aime/hate Juliet.
J(t) : Juliet aime/hate Romeo.

Premier cas :
Plus Juliet aime Romeo moins Romeo aime Juliet ; plus Romeo aime Juliet, plus Juliet aime Romeo. Cela se traduit dans le système d'équations différentielles :

$$\begin{cases} \frac{dR}{dt} = -a^2 J \\ \frac{dJ}{dt} = b^2 R \end{cases} \tag{2.10}$$

Visualizer R et J en fonction du temps (choisir a et b) et le diagramme R,J. Voir love_eq1 et rj1.

Deuxième cas : '"The cautious lovers"

$$\begin{cases} \frac{dR}{dt} = -a^2 R + b^2 J \\ \frac{dJ}{dt} = b^2 R - a^2 J \end{cases} \tag{2.11}$$

a^2 mesure la prudence, b^2 mesure la réactivité. Voir les solutions en fonction de a et b. Pour plotter les flèches voir : vectorfield.m et love_eq1_vect.m et love_eq2_vect.m.

Exercise : Resoudre l'oscillateur de Van der Pol [1]

$$\begin{cases} \dot{x} = y \\ \dot{y} = \mu(1 - x^2)y - x \end{cases} \tag{2.12}$$

1. http ://en.wikipedia.org/wiki/Van_der_Pol_oscillator

Plotter : $x(t)$, x vs y et x vs y en fonction de μ. (voir r_vdp.m et mu_vdp.m) . Suggestion : utiliser le solveur ode15s. Voir doc ode45 pour accuracy des méthodes.

Exercise : Etudier le système dynamique (modèle Lotka-Volterra proie-predateur)

$$\dot{x} = x(\alpha - cy) \tag{2.13}$$
$$\dot{y} = y(\gamma x - \delta) \tag{2.14}$$

avec α, c, γ et δ positifs. (Voire lot_volt_dyn.m et volterra.m).

Les termes ont la signification suivante :
– αx represent la croissance de la population des proies en absence des prédateurs.
– les termes $-cxy$ et $+\gamma xy$ sont l'interaction entre les espèces.
– le terme $-\delta y$ represent l'extinction des prédateurs en absence des proies.

2.2.2 Cinétique chimique

Cinetétique. Pour résoudre les équation de la cinétique chimique "phenomenologique", i.e. l'apparition/disparition des espèces en fonction du temps en partant des constant de réaction, il faut résoudre des systèmes d'équations différentielles. Cela est "difficile" à la main, en chimie souvent l'approximation de l'état stationnaire est employée. On prendra cette example pour le résoudre de façon approché, analytique et numérique. L'équation type est :

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C \tag{2.15}$$

On peut écrire pour les trois espèces :

$$\frac{d[A]}{dt} = -k_1[A] \tag{2.16}$$
$$\frac{d[B]}{dt} = k_1[A] - k_2[B] \tag{2.17}$$
$$\frac{d[C]}{dt} = k_2[B] \tag{2.18}$$

Les solutions analytiques sont :

$$[A] = [A]_0 e^{-k_1 t} \tag{2.19}$$

$$[B] = \begin{cases} [A]_0 \frac{k_1}{k_2 - k_1} \left(e^{-k_1 t} - e^{-k_2 t} \right); & k_1 \neq k_2 \\ [A]_0 \, k_1 t e^{-k_1 t} & \text{otherwise} \end{cases} \tag{2.20}$$

$$[C] = \begin{cases} [A]_0 \left(1 + \frac{k_1 e^{-k_2 t} - k_2 e^{-k_1 t}}{k_2 - k_1} \right); & k_1 \neq k_2 \\ [A]_0 \left(1 - e^{-k_1 t} - k_1 t e^{-k_1 t} \right); & \text{otherwise} \end{cases} \tag{2.21}$$

Dans l'approximation de l'état stationnaire, la dérivé de B par rapport au temps est fixée à zéro, donc :

$$\frac{d[B]}{dt} = 0 = k_1[A] - k_2[B] \tag{2.22}$$

$$[B] = \frac{k_1}{k_2}[A] \tag{2.23}$$

$$\frac{d[C]}{dt} = k_1[A] \tag{2.24}$$

$$[C] = [A]_0 \left(1 - e^{-k_1 t} \right) \tag{2.25}$$

On prendre comme exemple : $[A]_0 = 1$, $[B]_0 = [C]_0 = 0$ et $k_1 = 1$, $k_2 = 10$. Voir le script steady_num.c et la fonction steady.m.

Exercise. Etant donnée le système cinétique suivante :

$$A \underset{\beta_1}{\overset{k_1}{\rightleftharpoons}} B \xrightarrow{k_2} C \tag{2.26}$$

écrire un programme qui donne l'évolution de A, B et C en fonction du temps, pour différents condition initiales et valeurs des constantes cinétiques. (Voire kin_2.m)

2.2.3 TD : Logistic Map

La map logistique représente une loi de population [2].

$$x_{n+1} = r x_n (1 - x_n) \tag{2.27}$$

2. http ://fr.wikipedia.org/wiki/Suite_logistique

ce loi de population est à étudier dans le domaine 0, 4 du paramètre r. Tracer donc :

1. l'évolution de x par différent valeurs de r ;
2. pour une valeur donnée de r la map 2D (x_n vs x_{n+1}) ;
3. pour une valeur donnée de r la map 3D (x_n vs x_{n+1} vs x_{n+2}) ;
4. les valeurs d'équilibre de x en fonction de r ;

Voir ensemble le dernier point. (exemples dans la directory "Logistic")

En particulier on peut remarquer les régions suivantes du paramètre r :

- $0 \leq r \leq 1$: la population s'éteint. L'espèce finira par mourir, quelle que soit la population de départ. Autrement dit, $\lim_{+\infty} x_n = 0$.
- $1 \leq r \leq 3$: l'effectif de la population se stabilise. Si $1 \leq r \leq 2$, la population finit par se stabiliser autour de la valeur $\frac{r-1}{r}$, quelle que soit la population initiale. Si $2 \leq r \leq 3$, elle finit également par se stabiliser autour de $\frac{r-1}{r}$ après avoir oscillé autour pendant quelque temps. La vitesse de convergence est linéaire, sauf pour $r = 3$ où elle est très lente.
- $3 \leq r \leq 3,57$: l'effectif de la population oscille entre 2, 4, 8... valeurs (puissance de 2).
- $3,57 \leq r$: l'effectif de la population est chaotique, sauf exception. Vers $r = 3,57$, le chaos s'installe. Aucune oscillation n'est encore visible et de légères variations de la population initiale conduisent à des résultats radicalement différents. La plupart des valeurs au-delà de 3,57 présentent un caractère chaotique, mais il existe quelques valeurs isolées de r avec un comportement qui ne l'est pas. Par exemple vers 3,82, un petit intervalle de valeurs de r présente une oscillation entre trois valeurs et pour r légèrement plus grand, entre six valeurs, puis douze, etc. D'autres intervalles offrent des oscillations entre 5 valeurs, etc. Toutes les périodes d'oscillation sont présentes, là encore indépendamment de la population initiale.
- Au-delà de $r = 4$, la population quitte l'intervalle $[0, 1]$ et diverge quasiment pour toutes les valeurs initiales

2.3 Cours 3 – 20/01 – Transformé de Fourier

En Matlab la fonction fft(X) calcule la transformé de Fourier discrete (DFT) de X en utilisant l'algorithm FFT.

Voit les exemples : fourier1.m, fourier2.m et fourier_gaus.m

Matalb peut aussi faire les transformées de fourier symboliques, avec la commande "fourier"

```
syms x y
f = exp(-x^2);
fourier(f, x, y)
```

2.4 Cours 4 – 03/03 – I/O et Fonctions

Output :

save prova.mat : sauvegarde les datas

disp(var 'string') : écrit sur ecran

save prova.txt A –ASCII : sauvegarde sur un fichier ascii

dlmwrite ('prova.tst', A, ';') : sauvegared sur un fischier

Input :

A = input('Enter input A') : lit de la ligne de commande;

load prova.mat : recharche les datas

load input.txt : charge les données dans input.txt et le place dans la matrice input (voir aussi par bouton)

A = xlsread ('prova.xls') : charge les données excel

[A, B]= xlsread ('prova.xls') : les données numériques dans A et le texte dans B

dlmread('test.in') : lire à partir d'un fichier ascii

Example

```
M = magic(3)*pi
dlmwrite('myFile.txt',M,'delimiter','\t','precision',3)
type('myFile.txt')
```

TD : Integrals Calculer : $I = \frac{2}{\sqrt{\pi}} \int_0^1 e^{-x^2} dx$ et $J = \int_1^2 \frac{sin(x)}{x} dx$ par la méthode des trapèzes. Rappelle :

$$\int_a^b f(x)dx \cong h\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{i=n-1} f(a+ih)\right] \tag{2.28}$$

$$h = \frac{b-a}{n} \tag{2.29}$$

Hint : la commande **feval** évalue la valeur d'une fonction en un ou plusieurs poins : feval('donc',a) (voir : trap1.m et trap.m)

First option : un fichier trap1.m avec l'appele direct à une fonction

```
% fonction trap1.m
function s = trap1( a,b,n )
h=(b-a)/n;
x=a :h :b;
N=length(x);
y=fonc1(x);
t=(y(1)+y(N))/2;
s=t+sum(y(2 :N-1));
s=s*h;
end

% fonction fonc1.m
function y = fonc1(t )
y=exp(-t.^2).*(2/sqrt(pi));
end

% fonction fonc1.m
function y = fonc2( t )
y=sin(t)./t;
end
```

la commande trap1(0,1,1000) donne la valeur de l'intégrale I (0.8427) . Pour obtenir J il faut modifier le fichier trap1.m en replaçant fonc1(x) avec fonc2(x).

Pour éviter cela on peut utiliser la commande **feval** qui évalue la valeur d'une fonction en un ou plusieurs points. Le fichier trap.m est une modification du precedent, et il est en mesure de calculer l'intégrale d'une fonction quelconque, définie par la chaine de caractères phi.

```
% fonction trap.m
function s = trap( phi,a,b,n )
h=(b-a)/n ;
x=a :h :b ;
y=feval(phi,x) ;
N=length(x) ;
t=(y(1)+y(N))/2 ;
s=t+sum(y(2 :N-1)) ;
s=s*h ;
end
```

Pour obtenir les intégrales il faut donc taper : I =trap('fonc1',0,1,1000) et J=trap('fonc2',1,2,1000)

Voir après les fonctions trapz, quad or integral (help ou doc).

TD : sélectionner une note Sonner une note rentrée par choix d'input. Le son est donné par la fonction sound(y,F) ou y est une fréquence et Fs la fréquence d'échantillonnage en sortie (sampling frequency). Fs=8192 est la valeur standard. $y = \sin(2 * \pi * f * t)$

ou t=0 :1/Fs :time aver time en seconds et f est la frequency en Hz. For example pour les notes "centrales" : 'C4' f=261.63 ; 'C#4' f=277.18 ; 'D4' f=293.66 ; 'D#4' f=311.13 ; 'E4' f=329.63 ; 'F4' f=349.23 ; 'F#4' f=369.99 ; 'G4' f=392.00 ; 'G#4' f=415.30 ; 'A4' f=440.00 ; 'A#4' f=466.16 ; 'B4' f=493.88 ;

(voir play_diat.m)

TD : Faire sonner une petite mélodie La fonction wavwrite permet d'enregistrer la mélodie sous le format wav. (voir ex1.m et ex2.m)

Exercise : faire sonner une note en la choisissant par bouton. (envoyer : nsound.m et nota.m). Faire une échelle musicale.

http ://www.mathworks.fr/videos/mathematical-modeling-in-matlab-89428.html

2.5 Examples.

2.5.1 Conway's Life Game

L'universe du Game of Life set use grille à 2D composé par des cellules et chaque élément peut avoir deux état : vivant ou mort. Chaque cellule interagit avec ces huit voisins, qui sont les cellules adjacentes horizontalement, verticalement et sur la diagonale. A chaque pas, on a les transitions suivantes :

– Une cellule vivante qui a moins de deux voisins vivantes meurt ;

– Une cellule vivante qui a deux ou trois voisins vivantes vive

– Une cellule vivante avec plus de trois voisins vivantes meurt

– Une cellule morte avec trois voisins vivantes deviens une cellule vivante

Voir life_test.m dans le folder MonteCarlo. Introduire d'abord les instructions sur des matrices sparses. Constuire une matrice "sparse" 50x50 : X = sparse(50,50) ;

Voir aussi la commande spy et drawnow.

2.5.2 Monte Carlo sur modèle de Ising 1D

Le modèle de Ising represent un ensemble de particules avec spin up or down. L'Hamiltonian pour un système 1D est :

$$H(\sigma) = -J \sum_i \sigma_i \sigma_{i+1} - h \sum_i \sigma_i \tag{2.30}$$

with $\sigma_i = \pm 1$

1. Définir des conditions initiales
2. sélectionner un spin, l'inverser et calculer la différence en énergie
3. si $\Delta E < 0$ alors on garde la nouvelle configuration
4. si $\Delta E >$ on accepte la nouvelle configuration avec une probabilité $e^{-\beta \Delta E}$ (pour cela on extrait un nombre aléatoire η entre 0 et 1 et on accepte la configuration si $e^{-\beta \Delta E} > \eta$.

Voire commande : hist , dismat.
Exemple : sur un modèle 1D avec PBC (le dernier spin est en interaction avec le premier), plotter :

1. la distribution initiale ;
2. l'évolution de spins ;
3. l'évolution de l'énergie ;
4. la distribution finale ;
5. la magnétisation $M = \sum_i \sigma_i$ à chaque pas et la valeur moyen sans considérer une partie d'équilibration ;
6. le rapport d'acceptation.

Voir : Ising1D_PBC.m et en_PBC.m. Suggestion d'exercise : plot de M en fonction de T.

2.6 Cours 11 – 29/04. Presentation des mini-projets

Chapitre 3

Sujets

Ici une liste des sujets possibles pour l'examen.

3.1 Cinétique chimique

– Evolution temporelles des espèces pour un réseau cinétique définie, avec choix de constants cinétiques, des conditions initiales.

3.2 Life game

Coder le "jeux de la vie" (Game of Life). L'universe du Game of Life set use grille à 2D composé par des cellules et chaque élément peut avoir deux état : vivant ou mort. Chaque cellule interagit avec ces huit voisins, qui sont les cellules adjacentes horizontalement, verticalement et sur la diagonale. A chaque pas, on a les transitions suivantes :

– Une cellule vivante qui a moins de deux voisins vivantes meurt ;

– Une cellule vivante qui a deux ou trois voisins vivantes vive

– Une cellule vivante avec plus de trois voisins vivantes meurt

– Une cellule morte avec trois voisins vivantes deviens une cellule vivante

Inserer la taille, le nombres de "vivants", la configuration initiale (random ou entre des cas predefinie) etc ... et suivre l'évolution pour un nombre de pas à définir.

3.3 Monte Carlo

- Sur un modele Ising à une dimension
- Monte Carlo sur modèle de "Ising 2D". Proprietés moyennes en fonction de la température. E.g. traduire un code fortran en Matlab.

3.4 Dynamique Moléculaire

Etant donnée un Hamiltonien au choix, évolution temporelle (verlet) et report des propriété (espace des phases, fréquences etc ...)

1. Deux oscillateurs harmonique couplée
2. Système planétaire

3.5 Dosage

Etudier le dosage d'un diacide AH_2 (V_a, C_a) par de la potasse KOH (V_b, C_b). Les équation de laréaction de dosage sont :

$$AH_2 + OH^- \leftrightharpoons AH^- + H_2O \tag{3.1}$$

$$AH^- + OH^- \leftrightharpoons A^{2-} + H_2O \tag{3.2}$$

$$\tag{3.3}$$

avec K_1 et K_2 les constantes d'équilibre correspondantes. On souhaite représenter graphiquement l'évolution du pH de la solution en fonction du volume V_b de potasse versée. On souhaite représenter aussi l'évolution des concentrations des espèces AH_2, AH^- et A^{2-} en fonction du volume de potasse versée. Exemples de diacides :

acide maléique : $K_1 = 1.4210^{-2}$, $K_2 = 8.5710^{-7}$
acide oxalique : $K_1 = 5.9010^{-2}$, $K_2 = 6.4010^{-5}$
acide adipique : $K_1 = 3.7110^{-5}$, $K_2 = 3.8710^{-6}$

3.6 Map Logisitque

La map logistique représente une loi de population [1].

$$x_{n+1} = rx_n(1 - x_n) \tag{3.4}$$

ce loi de population est à étudier dans le domaine 0, 4 du paramètre r. Tracer donc :

1. l'évolution de x par différent valeurs de r ;
2. pour une valeur donnée de r la map 2D (x_n vs x_{n+1}) ;
3. pour une valeur donnée de r la map 3D (x_n vs x_{n+1} vs x_{n+2}) ;
4. les valeurs d'équilibre de x en fonction de r ;

En particulier on peut remarquer les régions suivantes du paramètre r :

- $0 \leq r \leq 1$: la population s'éteint. L'espèce finira par mourir, quelle que soit la population de départ. Autrement dit, $\lim_{+\infty} x_n = 0$.
- $1 \leq r \leq 3$: l'effectif de la population se stabilise. Si $1 \leq r \leq 2$, la population finit par se stabiliser autour de la valeur $\frac{r-1}{r}$, quelle que soit la population initiale. Si $2 \leq r \leq 3$, elle finit également par se stabiliser autour de $\frac{r-1}{r}$ après avoir oscillé autour pendant quelque temps. La vitesse de convergence est linéaire, sauf pour $r = 3$ où elle est très lente.
- $3 \leq r \leq 3,57$: l'effectif de la population oscille entre 2, 4, 8... valeurs (puissance de 2).
- $3,57 \leq r$: l'effectif de la population est chaotique, sauf exception. Vers $r = 3,57$, le chaos s'installe. Aucune oscillation n'est encore visible et de légères variations de la population initiale conduisent à des résultats radicalement différents. La plupart

1. http ://fr.wikipedia.org/wiki/Suite_logistique

des valeurs au-delà de 3,57 présentent un caractère chaotique, mais il existe quelques valeurs isolées de r avec un comportement qui ne l'est pas. Par exemple vers 3,82, un petit intervalle de valeurs de r présente une oscillation entre trois valeurs et pour r légèrement plus grand, entre six valeurs, puis douze, etc. D'autres intervalles offrent des oscillations entre 5 valeurs, etc. Toutes les périodes d'oscillation sont présentes, là encore indépendamment de la population initiale.

– Au-delà de $r = 4$, la population quitte l'intervalle $[0, 1]$ et diverge quasiment pour toutes les valeurs initiales

3.7 Orbitales

– Application de la méthode de Huckel
– Atome d'hydrogène ou hydrogenoide