**LISEZ-MOI**
**(Mise à jour: 12 février 2014)**

Ce fichier (Lisez-moi.pdf) est une compilation des commentaires qui décrivent les différents programmes faits en MATLAB proposés par les auteurs.

Par la suite, le terme « objet » doit être compris dans le sens de la « programmation orientée objet ». En outre, l'expression générique "objet physique" se réfère à un électron, un proton, un neutron, etc.

## I – Fonctions du dossier @marche

Les fonctions MATLAB du dossier @marche concerne seulement une marche d'énergie potentielle.

```
function obj=marche(varargin)
% MARCHE est le constructeur de l'objet informatique "marche d'énergie
%   potentielle".
%   Cette classe est destinée à stocker les caractéristiques
%   (propriétés) physiques d'une marche d'énergie potentielle.
%
%   obj=marche(varargin)
%   Appelé avec aucun argument, il retourne un objet vide.
%   Appelé avec un argument (un objet informatique marche), il retourne l'objet
%   informatique.
%   Appelé avec 6 arguments:
%       >> nom  =  nom de la marche
%       >> x    =  position de la marche sur l'axe des x
%       >> Epe  =  énergie potentielle en entrée
%       >> Eps  =  énergie potentielle en sortie
%       >> ke   =  vecteur d'onde en entrée
%       >> ks   =  vecteur d'onde en sortie

%       << obj  =  objet informatique marche
%
```

```
function display(obj)
% DISPLAY affiche à l'écran les propriétés physiques
%   d'une marche d'énergie potentielle
%       >> obj  = objet informatique marche
%
```

```
function value=get(obj,property)
% GET retourne la valeur de la propriété demandée de l'objet marche.
%
%   value=get(obj,property)
%       >> obj      = objet informatique marche
%       >> property = propriété de la marche
%       << value    = valeur de la propriété
%
```

```
function obj=set(obj,varargin)
% SET retourne un nouvel objet informatique marche obtenu en modifiant une ou
%   plusieurs valeurs des propriétés d'un ancien objet marche
%
%   obj=set(obj,propriété1,valeur1,propriété2, valeur2, ...)
%       >> obj          = objet informatique marche
%       >> propriété    = propriété de la marche
%       >> valeur       = valeur de la propriété
%   Propriété   =
%'Nom','Position','EnergiePotentielleEntree','EnergiePotentielleSortie',
%                'VecteurOndeEntree', 'VecteurOndeSortie'
```

```
%       << obj          = nouvel objet informatique marche
%

function mat=matrix(obj)
% MATRIX retourne la matrice de réflexion-réfraction d'une marche d'énergie
%   potentielle.
%
%   mat=matrix(obj)
%       >> obj  = objet informatique marche
%       << mat  = matrice de réflexion-réfraction de l'objet marche
%

function [r,tau,phir,phitau]=factrtau(obj)
% FACTRTAU retourne les facteurs de réflexion et de transmission en
%   termes de fonctions d'onde pour un objet physique interagissant avec une
%   marche d'énergie potentielle. Les phases de ces facteurs sont
%   également données entre 0 et 2*pi.
%
%   [r,tau,phir,phitau]=factrtau(obj)
%       >> obj      = objet informatique marche
%       << r        = facteur de réflexion
%       << tau      = facteur de transmission
%       << phir     = phase à la réflexion
%       << phitau   = phase à la transmission
%

function [R,T]=factRT(obj)
% FactRT retourne les facteurs de réflexion et de transmission
%   en courant de probabilité d'un objet physique interagissant avec une
%   marche d'énergie potentielle.
%
%   [R,T]=FactRT(obj)
%       >> obj  = objet informatique marche
%       << R    = facteur de réflexion en courant de probabilité
%       << T    = facteur de transmission en courant de probabilité
%

function [x,psi,rho]=psirho(obj,pas,xmin,xmax)
% PSIRHO calcule les fonctions d'onde et les densités de probabilité
%   associées à un objet physique interagissant avec une marche d'énergie
%   potentielle. Le calcul s'effectue sur un domaine de l'axe des x
%   compris entre xmin et xmax.
%
%   [x,psi,rho]=psirho(obj,pas,xmin,xmax)
%       >> obj  = objet informatique marche
%       >> pas  = incrément selon l'axe x
%       >> xmin = limite inférieure de l'axe x
%       >> xmax = limite supérieure de l'axe x
%       << x    = variable de position sur l'axe x
%       << psi  = fonction d'onde de l'objet physique
%       << rho  = densité de probabilité de l'objet physique
%

function graph(obj,x,rho)
% GRAPH représente graphiquement une marche d'énergie potentielle
%   ainsi que la densité de probabilité d'un objet physique interagissant avec
%   cette marche
%
%   graph(obj,x,varargin)
%       >> obj      = Objet informatique marche
%       >> x        = Variable de position sur l'axe x
%       >> rho      = Densité linéique de probabilité
%
```

```
function mat=translation(obj1,obj2)
% TRANSLATION retourne la matrice de translation d'un objet informatique marche
%  1 à un objet informatique marche 2. Il s'agit précisément d'une matrice
%  relative à une zone d'énergie potentielle constante.
%
%   mat=translation(obj1,obj2)
%       >> obj1 = objet informatique marche 1
%       >> obj2 = objet informatique marche 2
%       << mat  = matrice de translation de la marche 1 à la marche 2
%
```

## II – Fonctions pour l'étude de l'interaction d'un objet physique avec une énergie potentielle à une dimension de forme quelconque.

Pour l'étude de l'interaction d'un objet physique avec une energie potentielle à une dimension de forme quelconque, on divise cette dernière en énergie potentielles élémentaires, c'est-à-dire que l'on considère une succession de marches. Évidemment, ces marches son séparées (l'une de celle immédiatement antérieure) par des régions (longueur L) d'énergie potentielle uniforme et sont indéxées par le nombre entier i.

Par la suite, on appelle système l'ensemble {énergie potentielle-objet physique}.

```
function k=VectOnde(E,m,Ep)
% VectOnde retourne le vecteur d'onde k d'un objet physique à partir de son
%   énergie totale, de sa masse et des valeurs de l'énergie potentielle
%   Ep à laquelle elle est soumise.
%
%   k=VectOnde(E,m,Ep)
%       >> E    = Energie de l'objet physique (eV)
%       >> m    = Masse de l'objet physique (relative à la masse de l'électron)
%       >> Ep   = Energie(s) potentielle(s) Ep(i) (eV), (i=1,2,...)
%       << k    = Vecteur(s) d'onde k(i) (nm^-1)
%
```

```
function sys=Systeme(varargin)
% SYSTEME retourne une matrice où chaque cellule contient les
%   caractéristiques des sous-potentiels (marches, constantes)
%   constituant l'énergie potentielle totale ainsi que les vecteurs
%   d'onde (entrées et sorties) de l'objet physique étudié.
%   Cet ensemble {énergie potentielle - objet physique} constitue
%   le système d'étude.
%
%   sys=Systeme(varargin)
%       >> varargin     = sous-potentiels (nombre variable)
%       << sys          = Energie potentielle totale + Vecteurs d'onde
%
%   Les sous-potentiels doivent être écrits dans l'ordre où ils
%   sont rencontrés par l'objet physique.
%
```

```
function [sys,E]=Epot(M,xt,Ep,m,E)
%   EPOT permet de fabriquer le système {objet physique - énergie
%   potentielle}
%
%   [sys,E]=Epot(M,xt,Ep,m,E)
%               >> M    = nom des différentes marches
%               >> xt   = positions de chaque marche
%               >> Ep   = valeur des énergies potentielles
%               >> m    = masse de l'objet physique (rapportée à l'électron)
%               >> E    = valeurs de l'énergie de l'objet physique
%               << sys  = système {objet physique - énergie). Il y a autant
```

```
%               de systèmes que de valeurs de E.
%               << E    = valeurs de E renvoyées (on exclut en effet les
%               valeurs telles que E=Ep pour éviter le cas singulier k=0
%

function TES=matrix(sys)
% MATRIX retourne la matrice de transfert pour le système
% {énergie potentielle-objet physique}.
%
%   TES=matrix(sys)
%       >> sys = Énergie potentielle-objet physique
%       << TES = Matrice de transfert (2*2) entre les plans d'entrée E et de
%                sortie S
%

function [LdB,delta,E]=LambdaDB(E,m,Ep)
% LambdaDB retourne la longueur d'onde de De Broglie d'un objet physique
%   à partir de son énergie totale, de sa masse et de(s) différentes
%   valeurs de l'énergie potentielle Ep(i) à laquelle elle est soumise (i >= 1).
%   Il retourne aussi la profondeur de pénétration dans une région où
%   l'énergie potentielle est supérieure à l'énergie de l'objet physique.
%
%   [LdB,delta,E]=LambdaDB(E,m,Ep)
%       >> E     = Energie de l'objet physique (eV)
%       >> m     = Masse de l'objet physique (relative à la masse de l'électron)
%       >> Ep    = Énergie(s) potentielle(s) Ep(i) (eV)
%       << LdB   = Longueur d'onde de de Broglie (nm)
%       << delta = Profondeur de pénétration (nm)
%

function [r,tau,phir,phitau]=factrtau(sys,TES)
% FACTRTAU retourne les facteurs de réflexion et de transmission
%   en terme de fonctions d'onde pour un objet soumis à une
%   énergie potentielle quelconque composée de discontinuités (objet
%   marche). Les phases de ces facteurs sont également données
%   entre 0 et 2*pi.
%
%   [r,tau,phir,phitau]=factrtau(sys)
%       >> sys      = Énergie potentielle
%       << r        = Facteur de réflexion en fonctions d'onde
%       << tau      = Facteur de transmission en fonctions d'onde
%       << phir     = Phase à la réflexion
%       << phitau   = Phase à la transmission
%

function [R,T]=factRT(sys,TES)
% FactRT retourne les facteurs de réflexion et de transmission en
%   courant de probabilité pour une particule soumise à une énergie
%   potentielle quelconque composée de discontinuités (objet marche).
%
%   [R,T]=factRT(sys)
%       >> sys  = énergie potentielle
%       << R    = facteur de réflexion en courant de probabilité
%       << T    = facteur de transmission en courant de probabilité
%

function [x,psi,rho]=psirho(sys,pas,xmin,xmax,TES,E)
% PSIRHO calcule les fonctions d'onde et
%   les densités de probabilité d'un objet physique
%   interagissant avec une énergie potentielle.
%
%   [x,psi,rho]=psirho(sys,pas,xmin,xmax)
%       >> sys    = Système {objet physique-énergie potentielle}
%       >> pas    = incrément selon l'axe x
```

```
%       >> xmin   = limite inférieure du domaine graphique
%       >> xmax   = limite supérieure du domaine graphique
%       << x      = position sur l'axe x
%       << psi    = fonction d'onde de l'objet physique
%       << rho    = densité de probabilité de l'objet physique
%

function probabilite=normalisation(sys,E,rho,pas)
%   NORMALISATION permet de vérifier que la densité linéique de probabilité
%   est bien normalisée à 1.
%
%   normalisation(sys,E,rho,pas)
%       >> sys    = Système {Objet physique - Energie potentielle}
%       >> E      = Valeurs des énergies de l'objet physique
%       >> rho    = Densité linéique de probabilité
%       >> pas    = Pas d'intégration
%       << probabilité (qui doit être proche de 1)
%

function graph(sys,x,rho)
%   GRAPH permet de tracer graphiquement, sur un axe Ox, l'énergie
%   potentielle et la densité linéique de probabilité de l'objet physique.

%   graph(sys,x,rho)
%       >> sys    = Système {énergie potentielle - objet physique}
%       >> x      = Valeur de la variable spatiale
%       >> rho    = Densité linéique de probabilité
%

function graphE(sys,E,param)
% graphE trace l'énergie potentielle considérée ainsi qu'un
% paramètre (le plus intéressant étant le facteur de transmission en
% probabilité, T) en fonction des différentes valeurs de l'énergie.
%
%   graphE(sys,E,param)
%       >>  sys   = Système {énergie potentielle-objet physique}
%       >>  E     = Valeurs des énergies
%       >>  param = Paramètre à tracer.
%

function graphe3D(M,xt,Ep,m,E,para)
% graphe3D permet de tracer le facteur de transmission T en fonction de
% l'énergie et d'un paramètre au choix.
%
%   graphe3D(M,xt,Ep,m,E,para)
%       >> M      = Nom des marches
%       >> xt     = Position des marches
%       >> Ep     = Valeurs des énergies potentielles
%       >> m      = Masse de l'objet physique
%       >> E      = Valeurs des énergies de l'objet physique
%       >> para   = Valeurs du paramètre caractéristique de l'énergie
%       potentielle
%

function sysconf(sys,E)
% SYSCONF détermine les énergies des états liés d'un objet physique confiné
%  dans un puits d'énergie potentielle.
%  On propose aussi une représentation graphique du puits et de ces énergies.
%
%     sysconf(sys,E)
%           >> sys      = Système
%           >> E        = Énergie de l'objet physique
%
```

```
function [M,xt,Ep]=discretisation(x,Epc)
% DISCRETISATION transforme une énergie potentielle continue,
%   dépendente de la variable de position x, en une énergie
%   potentielle discrète constituée d'une succession de marches
%   d'énergie potentielle.
%
%   [M,xt,Ep]=discretisation(x,Epc)
%       >> x    = Variable de position
%       >> Epc  = Energie potentielle "continue"
%       << M    = Nom d'une marche d'énergie potentielle
%       << xt   = Position d'une marche (transition)
%       << Ep   = Energie potentielle discrète
%

function [sys,E]=OscHarm(L,Ep0,m,varargin)
% OscArm détermine les énergies des état liés d'un objet physique confiné dans
%  un puits harmonique d'énergie potentielle.
%
%  [sys,E]=OscArm(L,Ep0,C,m,varargin)
%                              -- varargin signifie plusieurs arguments d'entrée
%   >> L    = largeur (nm) du puits
%   >> Ep0  = Profondeur (eV) du puits (Ep0 < 0)
%   >> m    = Masse (par rapport à l'électron) de l'objet physique
%   >> E    = Valeur de l'énergie de l'objet physique
%

function [sys,E]=OscAnHarm(L,Ep0,C,m,varargin)
% OscAnArm détermine les énergies des état liés d'un objet physique confiné dans
%  un puits anharmonique (cubique) d'énergie potentielle
%
%  [sys,E]=OscAnArm(L,Ep0,C,m,varargin)
%                              -- varargin signifie plusieurs arguments d'entrée
%   >> L    = largeur (nm) du puits
%   >> Ep0  = Profondeur (eV) du puits
%   >> C    = Constante de la perturbation
%   >> m    = Masse de l'objet physique (par rapport à l'électron)
%   >> varargin     = Arguments d'entrée optionnels
%           > nx    = Nombre de point selon Ox (défaut : 100)
%           > nE    = Nombre de valeurs de l'énergie (défaut : 500)
%           > Emin  = Valeur minimale de l'énergie (défaut : min(Ep))
%           > Emax  = Valeur maximale de l'énergie (défaut : 0)
%

function [sys,E]=Morse(Ep0,x0,a,m,varargin)
% Morse détermine les énergies des état liés d'un objet physique confiné dans
%  une énergie potentielle de Morse.
%
%   [sys,E]=Morse(Ep0,x0,a,m,varargin)
%                              -- varargin signifie plusieurs arguments d'entrée
%   >> Ep0          = Profondeur (eV) du puits
%   >> x0           = Abscisse du fond du puits
%   >> a            = Constante caractéristique (m^{-1})
%   >> m            = Masse (par rapport à l'électron) de l'objet physique
%   >> varargin     = Arguments d'entrée optionnels
%           >> nx   = Nombre de points selon Ox (défaut : 200)
%           >> nE   = Nombre de valeurs de l'énergie (défaut : 1000)
%           >> Emin = Valeur minimale de l'énergie (défaut : min(Ep))
%           >> Emax = Valeur maximale de l'énergie (défaut : 0)
%

function [sys,E]=EpLin(pente,m,varargin)
% EpLin détermine les énergies des états liés d'un objet physique qui interagit
%   avec une énergie potentielle linéaire de pente a (Ep = a*x).
```

```
%
%   [sys,E]=EpLin(pente,m,varargin)
%                              -- varargin signifie plusieurs arguments d'entrée
%   >> pente        = Pente de l'énergie potentielle (J/m)
%   >> m            = Masse (par rapport à l'électron) de l'objet physique
%   >> varargin     = Arguments d'entrée optionnels
%          >> nx    = Nombre de points selon Ox (défaut : 100)
%          >> nE    = Nombre de valeurs de l'énergie (défaut : 1000)
%          >> Emin = Valeur minimale de l'énergie (défaut : min(Ep))
%          >> Emax = Valeur maximale de l'énergie (défaut : Ep(max(x)))
%

function [sys,E]=Hydrog(Z,l,m,varargin)
% Hydrog détermine les énergies des états liés d'un objet physique confiné
%  dans un ion hydrogenoïde .
%
%  [sys,E]=Hydrog(Z,l,m,varargin)
%                              -- varargin signifie plusieurs arguments d'entrée
%     >> Z         = Charge de l'ion hydrogénoïde
%     >> l         = Nombre quantique secondaire (l > 0)
%     >> m         = Masse de l'objet physique (par rapport à l'electron)
%     >> varargin  = Arguments d'entrée optionnels
%         >> nx    = Nombre de points sur l'axe Ox (défaut: 200)
%         >> nE    = Nombre de valeurs de l'énergie (défaut: 1000)
%         >> Emin  = Valeur minimale de l'énergie (défaut: min(Ep))
%         >> Emax  = Valeur maximale de l'énergie (défaut: 0)
%

function [sys,E,Ep]=Rampe(Epi,Epf,pente,m,varargin)
% Rampe calcule les probabilités de réflexion et de transmission
%   d'une rampe d'énergie potentielle et en offre une représentation
%   graphique en fonction de l'énergie de l'objet physique.
%
%     [sys,E,Ep]=Rampe(Epi,Epif,pente,m,varargin)
%                   -- varargin indique un nombre variable d'arguments d'entrée
%     >> pente      = Pente de l'énergie potentielle (J/m)
%     >> m          = Masse (par rapport à l'électron) de l'objet physique
%     >> varargin   = Arguments d'entrée optionnels
%           >> nx   = Nombre de points selon Ox (défaut : 100)
%           >> nE   = Nombre de valeurs de l'énergie (défaut : 1000)
%           >> Emin = Valeur minimale de l'énergie (défaut : min(Ep))
%           >> Emax = Valeur maximale de l'énergie (défaut : Ep(max(x)))
%

function sysper(sys,E,pas)
% SYSPER determine les bandes d'énergie permises (de conduction) et interdites
%  (de valence) d'un obejet physique qui interagit avec une énergie potentielle
%  periodique.
%  On propose aussi une représentation graphique de cette périodicité et de ces
%  bandes.
%
%     sysper(sys,E,period)
%           >> sys      = Système
%           >> E        = Énergie de l'objet physique
%           >> pas      = Periodicité spatiale (pas) de l'énergie potentielle
%


function [En,sysfin,Efin,TESfin]=affinage(sys,m,En,varargin)
%   AFFINAGE permet d'affiner la détermination de l'énergie d'un état confiné
%
%   [En,sysfin,Efin,TESfin]=affinage(sys,m,En,varargin)
%       >> sys          = Système {Objet physique - énergie potentielle}
```

```
%       >> m            = Masse de l'objet physique
%       >> En           = Energie de l'état confiné (à affiner)
%       >> varargin     = Arguments d'entrée optionnels
%           > alpha     = Facteur (<< 1) tel que 2*alpha*En est le nouvel
%                             intervalle énergétique centré sur En
%           > nE        = Nombre de valeurs de l'énergie
%           > Emin      = Energie la plus basse du nouvel intervalle énergétique
%           > Emax      = Energie la plus haute du nouvel intervalle énergétique
%       << En           = Valeur affinée de l'énergie de l'état confiné
%       << sysfin       = Nouveau système sur l'intervalle énergétique
%       << Efin         = Nouvelles valeurs affinées de l'énergie
%       << TESfin       = Matrice de transfert de sysfin
%


function [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
%  Schrodinger résoud l'équation de Schrödinger pour déterminer les valeurs
%  propres et les vecteurs propres, c'est-à-dire les énergies des états
%  propres (stationnaires).
%  Elle sert en particulier pour les états liés (confinés) car dans certains
%  cas la méthode de la matrice de transfert est trop sensible aux
%  approximations numériques.
%
%   [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
%
%     >> xmin     = Limite inférieure du domaine (nm)
%     >> xmax     = Limite supérieure du domaine (nm)
%     >> Ep       = Énergie potentielle (eV)
%     >> m        = Masse de l'objet physique (par rapport à l'électron)
%     >> num_sol  = Nombre de solutions (valeurs propres) souhaitées
%     << En       = Valeur propre (énergie état confiné) (eV)
%     << psin     = Fonction d'onde associée à la valeur propre En
%     << rhon     = Densité de probabilité associée à la valeur propre En
%
```