**README**
**(update: February 12th, 2014)**

This file (readme.pdf) is a collection of various comments which describe the MATLAB functions proposed by the authors.

Herein, the term "object" must be understood in the sense of the Object-Oriented programming. Furthermore, the generic expression "physical object" refers to an electron, a proton, a neutron, etc.

## I – Functions on folder @step

The MATLAB functions on the folder @step acts only on one step of potential energy.

```
function obj=step(varargin)
% STEP contains the physical characteristics (properties) of the object "step of
potential energy" (by the next called simply as "step").
%
%   obj=step(varargin) -- varargin states for various input arguments.
%   Without any argument, returns none.
%   With one argument (one object step), returns the object.
%   With 6 arguments:
%       >> name  =  Step name
%       >> x     =  Position of the step on the Ox axis
%       >> Epe   =  Input potential energy
%       >> Eps   =  Output potential energy
%       >> ke    =  Input wavevector
%       >> ks    =  Output wavevector
%
%       << obj   =  Step object
%

function display(obj)
% DISPLAY prints on the screen the physical properties of a step.
%
%   display(obj)
%       >> obj  = Step object
%

function value=get(obj,property)
% GET returns the value of a step property.
%
%   value=get(obj,property)
%       >> obj      = Step object
%       >> property = Step property
%
%   Property =
%'Name','Position','InputPotentialEnergy','OutputPotentialEnergy',
%                  'InputWaveVector', 'OutputWaveVector'
%
%       << value    = Property value
%

function obj=set(obj,varargin)
% SET returns a new step obtained by modifying one or some
%   values of the property of a previous step.
%
%   obj=set(obj,property1,value1,property2,value2,...)
%                             -- varargin states for various input arguments.
```

```
%       >> obj          = Step object
%       >> property     = Step property
%       >> value        = Property value
%
%   Property =
%'Name','Position','InputPotentialEnergy','OutputPotentialEnergy',
%                  'InputWaveVector', 'OutputWaveVector'
%
%       << obj          = New step object
%

function mat=matrix(obj)
% MATRIX computes the matrix which characterises a step of potential energy.
%
%   mat=matrix(obj)
%       >> obj  = Step object
%       << mat  = Matrix of the step
%

function [r,tau,phir,phitau]=factrtau(obj)
% FACTRTAU returns the factors of reflection and transmission in terms of
%   wavefunction complex amplitudes of a physical object which interacts
%   with a step of potential energy.
%   The phase (phi) of these complex factors are given between 0 and 2*pi.
%
%   [r,tau,phir,phitau]=factrtau(obj)
%       >> obj      = Step object
%       << r        = Factor of reflection
%       << tau      = Factor of transmission
%       << phir     = Phase at reflection
%       << phitau   = Phase at transmission
%

function [R,T]=factRT(obj)
% FactRT returns the reflection and transmission probabilities
%   of a physical object which interacts with a step of potential energy.
%
%   [R,T]=FactRT(obj)
%       >> obj  = Step object
%       << R    = Probability of reflection
%       << T    = Probability of transmission
%

function [x,psi,rho]=psirho(obj,gate_spacing,xmin,xmax)
% PSIRHO determines the wave function and the probability density of a
%   physical object which interacts with a step of potential energy.
%   Computing is made over an interval on Ox axis between xmin and xmax.
%
%   [x,psi,rho]=psirho(obj,gate_spacing,xmin,xmax)
%       >> obj          = Step object
%       >> gate_spacing  = Gate spacing on axis Ox
%       >> xmin         = Lowest limit on axis Ox
%       >> xmax         = Highest limit on axis Ox
%       << x            = Variable of position on Ox axis
%       << psi          = Wave function
%       << rho          = Probability density
%

function graph(obj,x,rho)
% GRAPH plots a step of potential energy and the probability density of a
%   physical object which interacts with the considered (drawn) step.
%
%   graph(obj,x,rho)
```

```
%       >> obj      = Step object
%       >> x        = Variable of position on Ox axis
%       >> rho      = Probability density
%

function mat=translation(obj1,obj2)
% TRANSLATION computes the matrix of propagation from a step to another.
%   This matrix is characteristic of a region (length L) of uniform potential
%   energy.
%
%   mat=translation(obj1,obj2)
%       >> obj1 = Step object 1
%       >> obj2 = Step object 2
%       << mat  = matrix of propagation from step 1 to step 2
%
```

## II – Functions for addressing interaction of a physical object with a one- dimensional potential energy of assorted shape.

In order to address the issue of the interaction of a physical object with an one-dimensional potential energy of assorted shape, this last one is divided into several elementary potential energies, i.e considered as a succession of steps. Obviously, these steps are separated by regions (length L) of uniform potential energy, each of them being indexed by the integer i.

By the next, the system is {potential energy - physical object}.

```
function k=WaveVect(E,m,Ep)
% WaveVect returns the complex wavevector k of a physical object using its
%  energy, its mass, and the values of the potential energy.
%
%   k=WaveVect(E,m,Ep)
%       >> E    = Physical object energy (eV)
%       >> m    = Physical object mass (with respect to electron mass)
%       >> Ep   = Potential energies Ep(i) (eV), (i=1,2,...)
%       << k    = Wavevectors k(i) (nm^-1)
%

function sys=System(varargin)
% System returns a matrix structure where each cell contains the characteristics
%     of the elementary potential energies which constitute the whole potential
%     energy, and the wavevectors (input and output) of the physical object.
%     The system is: {potential energy - physical object}.
%
%   sys=System(varargin) -- varargin states for various input arguments.
%       >> varargin     = Elementary potential energies (variable number)
%       << sys          = System
%
%   The elementary potential energies must be written following the order for
%     which they are "encountered" by the physical object.
%

function [sys,E]=Epot(M,xt,Ep,m,E)
% Epot permits to construct the system {potential energy - physical object}.
%
%   [sys,E]=Epot(M,xt,Ep,m,E)
%       >> M    = Step names
%       >> xt   = Step positions on Ox axis
%       >> Ep   = Potential energy values
%       >> m    = Physical object mass (with respect to electron mass)
%       >> E    = Physical object energy values
%       << sys  = System. Number of systems = number of energy values
```

```
%       << E    = E-values kept (E=Ep is excluded to avoid the singularity k=0)
%
function TES=matrix(sys)
% MATRIX returns the transfer matrix of the system.
%
%   TES=matrix(sys)
%       >> sys = System
%       << TES = Transfer matrix (2*2) of the system
%
function [LdB,delta,E]=LambdaDB(E,m,Ep)
% LambdaDB calculates the de Broglie wavelength of a physical object using its
%  energy, its mass, and the various potential energy values Ep(i) (i=1,2,...).
%   The penetration depth (delta) in a region where the potential energy is
%   greater than the physical object energy is also given.
%
%   [LdB,delta,E]=LambdaDB(E,m,Ep)
%       >> E        = Physical object energy (eV)
%       >> m        = Physical object mass (with respect to electron mass)
%       >> Ep       = Potential energy Ep(i) (eV)
%       << LdB      = De Broglie wavelength LdB(i) (nm)
%       << delta    = Penetration depth delta(i) (nm)
%       << E
%
function [r,tau,phir,phitau]=factrtau(sys,TES)
% FACTRTAU gives the factors of reflection and transmission in terms of
%   wave function complex amplitudes of a physical object which interacts
%   with a potential energy.
%   The phase (phi) of these complex factors are given between 0 and 2*pi.

%   [r,tau,phir,phitau]=factrtau(sys,TES)
%       >> sys      = System
%       >> TES      = Transfer matrix of the system
%       << r        = Factor of reflection
%       << tau      = Factor of transmission factor
%       << phir     = Phase shift at reflection
%       << phitau   = Phase shift at transmission
%
function [R,T]=factRT(sys,TES)
% FactRT returns the reflection and transmission probabilities of a physical
%   object which interacts with a potential energy.
%
%   [R,T]=FactRT(sys,TES)
%       >> sys  = System
%       >> TES  = Transfer matrix of the system
%       << R    = Probability of reflection
%       << T    = Probability of transmission
%
function [x,psi,rho]=psirho(sys,gate_spacing,xmin,xmax,TES,E)
% PSIRHO determines the wavefunction and the probability density of a
%   physical object which interacts with a potential energy.
%
%   [x,psi,rho]=PSIRHO(sys,pas,xmin,xmax)
%       >> sys                = System
%       >> gate_spacing       = Gate spacing on Ox-axis
%       >> xmin               = Lowest limit on Ox axis
%       >> xmax               = Highest limit on Ox axis
%       << x                  = Position on axis Ox
%       << psi                = Wave function complex amplitude
%       << rho                = Probability density
```

```
%

function probability=normalization(sys,E,rho,pas)
%   NORMALIZATION verifies that the probability density is normalized to 1.
%
%   probability=normalization(sys,E,rho,pas)
%       >> sys    = System
%       >> E      = Physical object energy values
%       >> rho    = Probability density
%       >> pas    = Integration step
%       << probability (must be closed to 1)
%

function graph(sys,x,rho)
% GRAPH plots the potential energy and the probability density of a
%   physical object.
%
%   graph(sys,x,rho)
%       >> sys    = System
%       >> x      = Spatial variable (coordinate on Ox axis)
%       >> rho    = Probability density
%


 function graphe3D(M,xt,Ep,m,E,para)
% graphe3D plots the transmission probability T versus energy E and an user-
%  chosen parameter.
%
%   graphe3D(M,xt,Ep,m,E,para)
%       >> M      = Step names
%       >> xt     = Step positions
%       >> Ep     = Potential energy values
%       >> m      = Physical object mass (with respect to electron mass)
%       >> E      = Physical object energy values
%       >> para   = Values of the parameter chosen by the user
%

function graphE(sys,E,param)
% graphE plots the considered potential energy and a parameter chosen by user
% (the most interesting is the transmission probability T) versus energy values.
%
%   graphE(sys,E,param)
%       >>  sys   = System
%       >>  E     = Physical object energy values
%       >>  param = Parameter considered
%


function sysconf(sys,E)
% sysconf determines the bound state energies of a physical object confined
%  in a well of potential energy.
%  Also, a graphical representation of the well and these energies is proposed.
%
%     sysconf(sys,E)
%           >> sys      = System
%           >> E        = Energy of the physical object
%


function [M,xt,Ep]=digitalization(x,Epc)
% DIGITALIZATION transforms a continuous potential energy to a digitalized
%   one made of various steps.
%
%   [M,xt,Ep]=digitalization(x,Epc)
%       >> x    = Coordinate on Ox axis
%       >> Epc  = Continuous potential energy
```

```
%       << M    = Name of a potential energy step
%       << xt   = Step position
%       << Ep   = Digitalized potential energy
%

function [sys,E]=OscHarm(L,Ep0,m,varargin)
% OscHarm returns the bound state energies of a physical object confined in an
%  harmonic well of potentiel energy
%
%  [sys,E]=OscHarm(L,Ep0,C,m,varargin)
%                              -- varargin states for various input arguments
%   >> L    = Well width (nm)
%   >> Ep0  = Well depth (eV) (Ep0 < 0)
%   >> m    = Physical object mass (with respect to electron mass)
%   >> E    = Physical object energy values
%

function [sys,E]=OscAnHarm(L,Ep0,C,m,varargin)
% OscAnHarm returns the bound state energies of a physical object confined in a
%  cubic anharmonic well of potential energy
%
%  [sys,E]=OscAnHarm(L,Ep0,C,m,varargin)
%                              -- varargin states for various input arguments
%     >> L         = Well width (nm)
%     >> Ep0       = Well depth (eV) (Ep0 < 0)
%     >> C         = Constant
%     >> m         = Physical object mass (with respect to mass electron)
%     >> varargin  = Optional input arguments
%           > nx   = Gate number on Ox axis (default value: 100)
%           > nE   = Number of energy values (default: 500)
%           > Emin = Lowest energy value (default: min(Ep))
%           > Emax = Highest energy value (default: 0)
%

function [sys,E]=Morse(Ep0,x0,a,m,varargin)
% MORSE gives the bound state energies of a physical object which interacts
%  with a Morse potential energy.
%
%   [sys,E]=Morse(Ep0,x0,a,m,varargin)
%                              -- varargin states for various input arguments
%     >> Ep0          = Well depth (eV)
%     >> x0           = Position of the well minimum
%     >> a            = Characteristic constant (m^{-1})
%     >> m            = Physical object mass (with respect to mass electron)
%     >> varargin     = Optional input arguments
%           >> nx     = Gate number on Ox (default value: 200)
%           >> nE     = Number of energy values (default: 1000)
%           >> Emin   = Lowest energy value (default: min(Ep))
%           >> Emax   = Highest energy value (default: 0)
%

function [sys,E]=EpLin(slope,m,varargin)
% EpLin determines the bound state energies of a physical object which interacts
%   with a linear potential energy of slope a (Ep = a*x).
%
%   [sys,E]=EpLin(slope,m,varargin)
%                              -- varargin states for various input arguments.
%   >> slope      = Slope of the potential energy (J/m)
%   >> m          = Mass of the physical object (with respect to electron mass)
%   >> varargin   = Optional input arguments:
%        >> nx    = Number of Ox-gates (default: 100)
%        >> nE    = Number of energie values (default: 1000)
%        >> Emin  = Lowest value of energy (default: min(Ep))
```

```
%        >> Emax  = Highest value of energy (default: Ep(max(x)))
%

function [sys,E]=Hydrog(Z,l,m,varargin)
% Hydrog determines the bound state energies of a physical object in an
%  hydrogenoic atom.
%
%  [sys,E]=Hydrog(Z,l,m,varargin) -- varargin states for various input arguments
%     >> Z         = Charge number of the hydrogenoic atom
%     >> l         = Secondary quantum number (l > 0)
%     >> m         = Mass of the physical object (with respect to mass electron)
%     >> varargin  = Optional input arguments
%         >> nx    = Gate number on Ox axis (default value: 200)
%         >> nE    = Number of energy values (default: 1000)
%         >> Emin  = Lowest energy value (default: min(Ep))
%         >> Emax  = Highest energy value (default: 0)
%

function sysper(sys,E,period)
% SYSPER determines the allowed energy bands (conduction bands) and the
%  forbidden ones (valence bands) of a physical object which interacts with a
%  periodic potential energy.
%  Also, a graphical representation of the potential energy and these energy
%  bands is proposed.
%
%     sysper(sys,E,period)
%           >> sys      = System
%           >> E        = Energy of the physical object
%           >> period   = Spatial period of the potential energy
%

function [sys,E,Ep]=ramp(Epi,Epf,slope,m,varargin)
% RAMP returns the reflection and transmission probabilities
%   for a ramp of potential energy with a given inclination and proposes a
%   graphical representation versus the energy of the physical object.
%
%     [sys,E,Ep]=ramp(Epi,Epif,pendiente,m,varargin)
%                             -- varargin states for various input arguments
%     >> slope          = Slope of the ramp of potential energy (J/m)
%     >> m              = Physical object mass (with respect to electron mass)
%     >> varargin       = Optional input arguments
%           >> nx       = Gate number on Ox axis (default value: 100)
%           >> nE       = Number of energy values (default: 1000)
%           >> Emin     = Lowest energy value (default: min(Ep))
%           >> Emax     = Highest energy value (default: Ep(max(x)))
%

function [En,sysfin,Efin,TESfin]=refine(sys,m,En,varargin)
%   REFINE determines the energy of a bound states more precisely.
%
%   [En,sysfin,Efin,TESfin]=refine(sys,m,En,varargin)
%                             -- varargin states for various input arguments
%       >> sys          = System
%       >> m            = Physical object mass
%       >> En           = Bound state energy (to refine)
%       >> varargin     = Optional input arguments
%            > alpha    = Factor (<< 1) such that 2*alpha*En be the new
                              energetic interval centered on En
%           > nE        = Number of energy values
%           > Emin      = Lowest energy of the new energetic interval
%           > Emax      = Highest energy of the new energetic interval
%       << En           = Refined value of the bound state energy
%       << sysfin       = New system over the new energetic interval
```

```
%       << Efin          = New refined energy values
%       << TESfin        = Transfer matrix of sysfin
%

function [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
% Solve Schrödinger's equation for energy eigenvalues and eigenfunctions,
%  i.e. energies of stationnary states.
%  It can be useful, especially for bound (o confined) states since, sometimes,
%  the transfer-matrix method is too sensible to the numerical approximations.
%
%   [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
%
%     >> xmin     =     Lowest value of x on Ox axis (nm)
%     >> xmax     =     Highest value of x on Ox axis (nm)
%     >> Ep       =     Potential energy (eV)
%     >> m        =     Physical object mass (with respect to the electron mass)
%     >> num_sol  =     Number of eigenvalues (energies) wanted
%     << En       =     Bound state energies (eV)
%     << psin     =     Wave function which corresponds to to En
%     << rhon     =     Probability density which corresponds to En
%
```