LÉEME
(Puesto al día: 12 de febrero de 2014)

Este fichero (léeme.pdf) es una recopilación de los comentarios que describen los distintos programas hechos en MATLAB propuestos por los autores.

A continuación, el término "objeto" tiene que ser entendido en el sentido de la "programación orientada a objetos". Asimismo, la expresión genérica "objeto físico" se refiere a un electrón, un protón, un neutrón, etc.

I – Funciones del archivo @escalón

Las funciones MATLAB en el archivo @escalón conciernen solamente a una energía potencial en forma de escalón.

```
function obj=escalón(varargin)
% ESCALÓN contiene las características (propiedades) físicas del objeto "escalón
%   de energía potencial" (en adelante llamado sencillamente "escalón").
%
%   obj=ESCALÓN(varargin) -- varargin indica varios argumentos de entrada.
%   Sin argumento, el objeto no existe (vacío).
%   Con un solo argumento (un objeto escalón), da el objeto.
%   Con 6 argumentos:
%       >> nombre = Nombre del escalón
%       >> x      = Posición del escalón en el eje Ox
%       >> Epe    = Energía potencial de entrada
%       >> Eps    = Energía potencial de salida
%       >> ke     = Vector de onda de entrada
%       >> ks     = Vector de onda de salida
%
%       << obj    =  Objeto escalón
%

function imprimir(obj)
% IMPRIMIR imprime por pantalla las propiedades físicas de un escalón.
%
%   IMPRIMIR(obj)
%       >> obj  = Objeto escalón
%

function valor=get(obj,propiedad)
% GET da el valor de una propiedad de un escalón.
%
%   valor=GET(obj,propiedad)
%       >> obj       = Objeto escalón
%       >> propiedad = Propiedad del escalón
%
%   Propiedad =
%   'Nombre','Posición','EnergíaPotencialEntrada','EnergíaPotencialSalida',
%                'VectorOndaEntrada', 'VectorOndaSalida'
%
%       << valor    = Valor de la propiedad
%

function obj=set(obj,varargin)
% SET da un nuevo objeto obtenido con la modificación de uno o varios valores de
%   las propiedades de un escalón.
%
%   obj=SET(obj,propiedad1,valor1,propiedad2,valor2,...)
%                            -- varargin indica varios argumentos de entrada.
%       >> obj          = Objet escalón
```

```
%       >> propiedad    = Propiedad del escalón
%       >> valor        = Valor de la propiedad
%
%   Propiedad =
%   'Nombre','Posición','EnergíaPotencialEntrada','EnergíaPotencialSalida',
%                'VectorOndaEntrada', 'VectorOndaSalida'
%
%       << obj          = Nuevo objeto escalón
%

function mat=matriz(obj)
% MATRIZ calcula la matriz característica de un escalón de energía potencial.
%
%   mat=MATRIZ(obj)
%       >> obj  = Objeto escalón
%       << mat  = Matriz del escalón
%

function [r,tau,phir,phitau]=factrtau(obj)
% FACTRTAU da los factores de reflexión y de transmisión en términos de las
%   amplitudes complejas de la función de onda de un objeto físico que
%   interactúa  con un escalón de energía potencial.
%   Las fases (phi) de estos tres factores complejos se sitúan entre 0 y 2*pi.
%
%   [r,tau,phir,phitau]=FACTRTAU(obj)
%       >> obj      = Objeto escalón
%       << r        = Factor de reflexión
%       << tau      = Factor de transmisión
%       << phir     = Desfase por la reflexión (fase del factor r)
%       << phitau   = Desfase por la transmisión (fase del factor tau)
%

function [R,T]=factRT(obj)
% FactRT da las probabilidades de reflexión y de transmisión de un objeto
%   físico que interactúa con un escalón de energía potencial.
%
%   [R,T]=FactRT(obj)
%       >> obj  = Objeto escalón
%       << R    = Probabilidad de reflexión
%       << T    = Probabilidad de transmisión
%

function [x,psi,rho]=psirho(obj,gate_spacing,xmin,xmax)
% PSIRHO determina la función de onda y la densidad lineal de probabilidad de
%   un objeto físico que interactúa con un escalón de energía potencial.
%   El cálculo se hace sobre un intervalo en el eje Ox entre xmin y xmax.
%
%   [x,psi,rho]=PSIRHO(obj,incremento,xmin,xmax)
%       >> obj          = Objeto escalón
%       >> incremento   = Incremento en el eje Ox
%       >> xmin         = Límite inferior en el eje Ox
%       >> xmax         = Límite superior en el eje Ox
%       << x            = Posición en el eje Ox
%       << psi          = Función de onda del objeto físico
%       << rho          = Densidad lineal de probabilidad del objeto físico
%

function graf(obj,x,rho)
% GRAF ofrece una representación gráfica del escalón de energía potencial y de
%   la densidad lineal de probabilidad de un objeto físico que interactúa
%   con el escalón (dibujado) considerado.
%
%   GRAF(obj,x,rho)
```

```
%       >> obj      = Objeto escalón
%       >> x        = Posición en el eje Ox
%       >> rho      = Densidad lineal de probabilidad
%

function mat=translación(obj1,obj2)
% TRANSLACIÓN calcula la matriz de propagación desde un escalón a otro.
%   Esta matriz caracteriza una zona (longitud L) de energía potencial uniforme.
%
%   mat=TRANSLATION(obj1,obj2)
%       >> obj1 = Objeto escalón 1
%       >> obj2 = Objeto escalón 2
%       << mat  = Matriz de propagación desde el escalón 1 hasta el escalón 2
%
```

II – Funciones para investigar la interacción del objeto físico con una energía potencial de dimensión una y de cualquier forma.

Para investigar la interacción de un objeto físico con una energía potencial de una dimensión y de cualquier forma, esta última está dividida en energías potenciales elementales, es decir considerada como una sucesión de escalones. Obviamente,esos escalones están separados (uno del inmediatamente anterior) por regiones (longitud L) de energía potencial uniforme, siendo cada una indexada por el número entero i.

En adelante, llamamos sistema al conjunto {energía potencial – objeto físico}.

```
function k=VectOnda(E,m,Ep)
% VectOnda da el vector de onda complejo k de un objeto físico utilizando su
%   energía, su masa, y los valores de la energía potencial.
%
%   k=VectOnda(E,m,Ep)
%       >> E    = Energía del objeto físico (eV)
%       >> m    = Masa del objeto físico (respecto a la del electrón)
%       >> Ep   = Energías potenciales Ep(i) (eV), (i=1,2,...)
%       << k    = Vectores de onda k(i) (nm^-1)
%

function sys=Sistema(varargin)
% Sistema da una estructura (matriz) cuyas celdas contienen, cada una, las
%   características de las energías potenciales elementales que forman la
%   energía potencial total, y los vectores de onda (entrada y salida) del
%   objeto físico.
%   El sistema es: {energía potencial – objeto físico}.
%
%   sys=Sistema(varargin) -- varargin indica varios argumentos de entrada.
%       >> varargin     = Energías potenciales elementales (número variable)
%       << sys          = Sistema
%
% Las energías potenciales elementales tienen que ser escritas siguiendo el orden
% en el que el que se encuentran, progresando del emisor hasta el receptor.
%

function [sys,E]=Epot(M,xt,Ep,m,E)
%   Epot permite construir el sistema {energía potencial – objeto físico}.
%
%   [sys,E]=Epot(M,xt,Ep,m,E)
%       >> M    = Nombres de los escalones
%       >> xt   = Posición de los escalones en el eje Ox
%       >> Ep   = Valores de la energía potencial
%       >> m    = Masa del objeto físico (respecto a la del electrón)
%       >> E    = Valores de la energía del objeto físico
%       << sys  = Sistema. Número de sistemas = número de valores de energía
```

```
%       << E    = Valores de E guardados (E=Ep está excluido para evitar la
%                 singularidad k=0)
%

function TES=matriz(sys)
% MATRIZ calcula la matriz de transferencia del sistema.
%
%   TES=matriz(sys)
%       >> sys = Sistema
%       << TES = Matriz de transferencia (2*2) del sistema
%

function [LdB,delta,E]=LambdaDB(E,m,Ep)
% LambdaDB calcula la longitud de onda de De Broglie de un objeto físico
%  utilizando su energía, su masa, y los varios valores de la energía potencial
%  Ep(i) (i=1,2,...).
%  También LambdaDB calcula la profundidad de penetración (delta) en una región
%  donde la energía potencial es más grande que la del objeto físico.
%
%   [LdB,delta,E]=LambdaDB(E,m,Ep)
%       >> E        = Energía del objeto físico (eV)
%       >> m        = Masa del objeto físico (respecto a la del electrón)
%       >> Ep       = Energía potencial Ep(i) (eV)
%       << LdB      = Longitud de onda de De Broglie LdB(i) (nm)
%       << delta    = Profundidad de penetración delta(i) (nm)
%       << E
%

function [r,tau,phir,phitau]=factrtau(sys,TES)
% FACTRTAU da los factores de reflexión y de transmisión en términos de las
%   amplitudes complejas de la función de onda de un objeto físico que
%   interactúa  con un escalón de energía potencial.
%   Las fases (phi) de estos tres factores complejos se sitúan entre 0 y 2*pi.
%
%   [r,tau,phir,phitau]=FACTRTAU(sys,TES)
%       >> sys      = Sistema
%       >> TES      = Matriz de transferencia del sistema
%       << r        = Factor de reflexión
%       << tau      = Factor de transmisión
%       << phir     = Desfase por la reflexión (fase de r)
%       << phitau   = Desfase por la transmisión (fase de tau)
%

function [R,T]=factRT(sys,TES)
% FactRT da las probabilidades de reflexión y de transmisión de un objeto
%   físico que interactúa con un escalón de energía potencial.
%
%   [R,T]=FactRT(sys,TES)
%       >> sys  = Sistema
%       >> TES  = Matriz de transferencia del sistema
%       << R    = Probabilidad de reflexión
%       << T    = Probabilidad de transmisión
%

function [x,psi,rho]=psirho(sys,incremento,xmin,xmax,TES,E)
% PSIRHO determina la función de onda y la densidad lineal de probabilidad de
%   un objeto físico que interactúa con una energía potencial.
%
%   [x,psi,rho]=PSIRHO(sys,pas,xmin,xmax)
%       >> sys          = Sistema
%       >> incremento   = Incremento en el eje Ox
%       >> xmin         = Límite inferior de x en el eje Ox
%       >> xmax         = Límite superior de x en el eje Ox
%       << x            = Posición en el eje Ox
```

```
%       << psi    = Amplitud compleja de la función de onda del objeto físico
%       << rho    = Densidad lineal de probabilidad del objeto físico
%

function probabilidad=normalización(sys,E,rho,inc_int)
%   NORMALIZACIÓN permite comprobar que la densidad lineal de probabilidad está
%   normalizada a 1.
%
%   probabilidad=NORMALIZACIÓN(sys,E,rho,pas)
%       >> sys          = Sistema
%       >> E            = Valores de energía del objeto físico
%       >> rho          = Densidad lineal de probabilidad
%       >> inc_int      = Incremento de integración
%       << probabilidad (tiene que ser cercana a 1)
%

function graf(sys,x,rho)
%   GRAF ofrece una representación gráfica de la energía potencial y de la
%   densidad lineal de probabilidad de un objeto físico que interactúa con
%   la misma.
%
%   GRAF(sys,x,rho)
%       >> sys    = Sistema
%       >> x      = Coordenada en el eje Ox
%       >> rho    = Densidad lineal de probabilidad
%


function graf3D(M,xt,Ep,m,E,para)
% graf3D ofrece una representación gráfica de la probabilidad de transmisión T
%  en función de la energíe E y de otro parámetro elegido por el usuario.
%
%   graf3D(M,xt,Ep,m,E,para)
%       >> M      = Nombres de los escalones
%       >> xt     = Posiciones de los escalones
%       >> Ep     = Valores de la energía potencial
%       >> m      = Masa del objeto físico (respecto a la del electrón)
%       >> E      = Valores de la energía del objeto físico
%       >> para   = Valores del parámetro elegido por el  usuario
%


function grafE(sys,E,param)
% grafE representa gráficamente la energía potencial considerada y un parámetro
%  (el más interesante es la probabilidad de transmisión T) en función de los
%   valores de la energía.
%
%   grafE(sys,E,param)
%       >>  sys   = Sistema
%       >>  E     = Valores de la energía del objeto físico
%       >>  param = Parámetro considerado
%


function sisconf(sys,E)
% SISCONF determina las energías de los estados ligados de un objeto físico
%  confinado en un pozo de energía potencial.
%  También se ofrece una representación gráfica del pozo y de esas energías.
%
%     SISCONF(sys,E)
%           >> sys      = Sistema
%           >> E        = Energía del objeto físico
%


function [M,xt,Ep]=digitalización(x,Epc)
% DIGITALIZATION transforma una energía potencial continua en una energía
```

```
%   potencial digitalizada, es decir formada por varios escalones.
%
%   [M,xt,Ep]=DIGITALIZATION(x,Epc)
%       >> x    = Coordenada en el eje Ox
%       >> Epc  = Energía potencial continua
%       << M    = Escalones de energía potencial
%       << xt   = Posición de los escalones
%       << Ep   = Energía potencial digitalizada
%

function [sys,E]=OscArm(L,Ep0,m,varargin)
% OscArm determina las energías de los estados ligados de un objeto físico
%  confinado en un pozo armónico de energía potencial.
%
%  [sys,E]=OscArm(L,Ep0,C,m,varargin)
%                            -- varargin indica varios argumentos de entrada
%   >> L    = Anchura del pozo (nm)
%   >> Ep0  = Profundidad del pozo (eV) (Ep0 < 0)
%   >> m    = Masa del objeto físico (respecto a la del electrón)
%   >> varargin         = Argumentos de entrada opcionales
%           > nx        = Número de puntos en el eje Ox (defecto: 100)
%           > nE        = Número de valores de energía (defecto: 500)
%           > Emin      = Valor inferior de la energía (defecto: min(Ep))
%           > Emax      = Valor superior de la energía (defecto: 0)
%

function [sys,E]=OscAnArm(L,Ep0,C,m,varargin)
% OscAnArm determina las energías de los estados ligados de un objeto físico
%  confinado en un pozo anarmónico (cúbico) de energía potencial
%
%  [sys,E]=OscAnArm(L,Ep0,C,m,varargin)
%                            -- varargin indica varios argumentos de entrada
%     >> L         = Anchura del pozo (nm)
%     >> Ep0       = Profundidad del pozo (eV) (Ep0 < 0)
%     >> C         = Constante
%     >> m         = Masa del objeto físico (respecto a la del electrón)
%     >> varargin  = Argumentos de entrada opcionales
%           > nx   = Número de puntos en el eje Ox (defecto: 100)
%           > nE   = Número de valores de energía (defecto: 500)
%           > Emin = Valor inferior de la energía (defecto: min(Ep))
%           > Emax = Valor superior de la energía (defecto: 0)
%

function [sys,E]=Morse(Ep0,x0,a,m,varargin)
% MORSE determina las energías de los estados ligados de un objeto físico que
%  interactúa con una energía potencial de Morse.
%
%   [sys,E]=Morse(Ep0,x0,a,m,varargin)
%                            -- varargin indica varios argumentos de entrada
%     >> Ep0          = Profundidad del pozo (eV)
%     >> x0           = Posición del mínimo del pozo energético
%     >> a            = Constante característica (m^{-1})
%     >> m            = Masa del objeto físico (respecto a la del electrón)
%     >> varargin     = Argumentos de entrada opcionales
%           >> nx     = Número de puntos en el eje Ox (defecto: 200)
%           >> nE     = Número de valores de energía (defecto: 1000)
%           >> Emin   = Valor inferior de la energía (defecto: min(Ep))
%           >> Emax   = Valor superior de la energía (defecto: 0)
%

function [sys,E]=EpLin(pendiente,m,varargin)
% EpLin determina las energías de los estados ligados de un objeto físico que
%   interactúa con una energía potencial lineal de pendiente a (Ep = a*x).
```

```
%
%   [sys,E]=EpLin(pendiente,m,varargin)
%                           -- varargin indica varios argumentos de entrada.
%   >> pendiente  = Pendiente de la energía potencial (J/m)
%   >> m          = Masa del objeto físico (respecto a la del electrón)
%   >> varargin   = Argumentos de entrada opcionales:
%        >> nx    = Número de puntos en el eje Ox (defecto: 100)
%        >> nE    = Número de valores de energía (defecto: 1000)
%        >> Emin  = Valor inferior de la energía (defecto: min(Ep))
%        >> Emax  = Valor superior de la energía (defecto: Ep(max(x)))
%

function [sys,E]=Hydrog(Z,l,m,varargin)
% Hydrog determina las energías de los estados ligados de un objeto físico
%  confinado en la energía potencial de un ión hidrogenoide .
%
%  [sys,E]=Hydrog(Z,l,m,varargin)
%                              -- varargin indica varios argumentos de entrada
%     >> Z         = Número de carga del ión hidrogenoide
%     >> l         = Número cuántico secundario (l > 0)
%     >> m         = Masa del objeto físico (respecto a la del electrón)
%     >> varargin  = Argumentos de entrada opcionales
%         >> nx    = Número de bines en el eje Ox (defecto: 200)
%         >> nE    = Número de valores de energía (defecto: 1000)
%         >> Emin  = Valor inferior de la energía (defecto: min(Ep))
%         >> Emax  = Valor superior de la energía (defecto: 0)
%

function sisper(sys,E,period)
% SISPER determina las bandas de energía permitidas (de conducción)
%  y prohibidas (de valencia) de un objeto físico que interactúa
%  con una energía potencial periódica.
%  También se ofrece una representación gráfica de esta periodicidad
%  y de esas bandas.
%
%     SISPER(sys,E,period)
%           >> sys      = Sistema
%           >> E        = Energía del objeto físico
%           >> period   = Periodicidad espacial de la energía potencial
%

function [sys,E,Ep]=Rampa(Epi,Epf,pendiente,m,varargin)
% Rampa calcula las probabilidades de reflexión y de transmissión
%   de una rampa de energía potencial y ofrece una representación
%   gráfica en función de la energía del objeto físico.
%
%     [sys,E,Ep]=Rampa(Epi,Epif,pendiente,m,varargin)
%                 -- varargin indica un número variable de argumentos de entrada
%     >> pendiente      = Pendiente de la energía potencial (J/m)
%     >> m              = Masa (respecto a la del electron) del objeto físico
%     >> varargin       = Argumentos de entrada opcionales
%           >> nx       = Número de puntos en el eje Ox (defecto: 100)
%           >> nE       = Número de valores de la energía (defecto: 1000)
%           >> Emin     = Valor mínimo de la energía (defecto: min(Ep))
%           >> Emax     = Valor máximo de la energía (defecto: Ep(max(x)))
%

function [En,sysfin,Efin,TESfin]=refine(sys,m,En,varargin)
%   REFINE determina con más precisión la energía de un estado ligado %
%     (confinado)
%
%   [En,sysfin,Efin,TESfin]=REFINE(sys,m,En,varargin)
%                             -- varargin indica varios argumentos de entrada
```

```
%       >> sys          = Sistema
%       >> m            = Masa del objeto físico (respecto a la del electrón)
%       >> En           = Energía para determinar con precisión
%       >> varargin     = Argumentos de entrada opcionales
%            > alpha    = Factor (<< 1) para que 2*alpha*En sea el nuevo
                              intervalo energético centrado en En
%            > nE       = Número de valores de energía
%            > Emin     = Valor inferior de la energía del nuevo intervalo
%            > Emax     = Valor superior de la energía del nuevo intervalo
%       << En           = Valor refinada de la energíe del estado ligado
%       << sysfin       = Nuevo sistema relativo al nuevo intervalo
%       << Efin         = Valores de la energía en el nuevo intervalo
%       << TESfin       = Matriz de transferencia de sysfin
%


function [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
%  Schrodinger resuelve la ecuación de Schrödinger para determinar sus valores
%  proprios y sus vectores propios, es decir las energías de los estados
%  proprios (estacionario).
%  Sirve en particular para los estados ligados (confinados) puesto que en
%  ciertos casos el método de la matriz de transferencia es demasiado sensible a
%  las aproximaciones numéricas.
%
%   [En,psin,rhon]=Schrodinger(xmin,xmax,Ep,m,num_sol)
%
%     >> xmin     =     Límite inferior en el eje Ox (nm)
%     >> xmax     =     Límite superior en el eje Ox (nm)
%     >> Ep       =     Energía potencial (eV)
%     >> m        =     Masa del objeto físico (respecto a la del electrón)
%     >> num_sol  =     Número de soluciones (energías) deseadas
%     << En       =     Energía del estado ligado (eV)
%     << psin     =     Función de onda correspondiente a En
%     << rhon     =     Densidad lineal de probabilidad correspondiente a En
%
```