Ejercicios Tema 2

Dia 1

Escribir una funcion lorentzian.m que devuelva una funcion lorenciana
(una representacion de la funcion delta, cuando a->0) de la forma
y(x) = (a/pi) / (a^2+x^2), con la siguiente interfaz

```
function y = lorentzian(a,x)
% Returns a normalized lorentzian
%   y(x) = (a/pi) / (a^2+x^2)
% Input:
%   a    : lorentzian width
%   x    : x value at which y(x) is required
% Output:
%   y    : value of the lorentzian at point x
% Usage:
%   f = lorentzian(0.5,3)
% J.M.Soler. Sept.2010
```

Hacer 'clear all' y calcular la funcion para varios valores de a y de x.

Modificar la funcion anterior para que admita que 'a' sea un vector columna,
'x' sea un vector fila, e 'y' sea una matriz para todos los valores de
'a' y 'x'. Modificar apropiadamente la documentacion de la cabecera.

Escribir un script llamado plotLorentzianFunction.m que dibuje la funcion
lorentzian(a,x) para a=(1:5) y x entre -20 y 20.

En un fichero simpleMatrixOperations.m:

Crear dos matrices 3x3 A y B, de numeros enteros.

Crear un vector fila row y un vector columna col, ambos enteros.

Multiplicar row*col, col*row, row*A, A*col y A*B,
comprobando cada resultado.

Multiplicar col'*row', row'*col', A'*row', col'*A' y B'*A', comprobando
que son los transpuestos de los anteriores. ¿Por que?

Obtener A(2,3), A(2,:), A(:,3) y A(1:2,2:3) y comprobar los resultados.

Obtener [A,B], [A;B], [A,B(:,2:3)], [A(1,:);B] y comprobar los resultados.

Usando un comando de una sola linea, crear una matriz x(10,3) con tres columnas
que contengan los diez primeros numeros naturales, sus cuadrados, y sus cubos.

Usando el comando save, guardar la variable x en un fichero binario xfile.mat

Usando fopen, fprintf y fclose, escribir la matriz x en un fichero de texto
xfile.txt

Usando el comando ls, comprobar que se han creado los ficheros xfile.mat y
xfile.txt

Tras hacer clear all, y usando el comando load, leer la matriz x, primero
de xfile.mat y despues de xfile.txt

Crear una matriz A sencilla 2x2 e invertirla a mano.

Comprobar que Ainv=inv(A) es la matriz inversa, y que A*Ainv=Ainv*A=I

Crear un vector columna b de dos elementos y resolver la ecuacion A*x=b con
tres metodos:

```
  x=Ainv*b
  x=A\b
  x=(b'/A')'
```

Comprobar que efectivamente se cumple la ecuacion.

Escribir una funcion que calcule la matriz de rotacion de un vector columna en 2D por un angulo phi (tal que rotXY=rotMat*XY), con la siguiente interfaz:

```
function R = rotMat( phi )
% Returns the 2x2 matrix that rotates any 2D column vector by angle phi:
%   [rotX;rotY] = R*[X;Y] with
%   rotX = X*cos(phi)-Y*sin(phi)
%   rotY = X*sin(phi)+Y*cos(phi)
% Input:
%   phi  : rotation angle (counter clockwise, in rad)
% Output:
%   R    : rotation matrix
% Usage:
%   R = rotMat( pi/4 )
%   newVec = R*vec
```

En la linea de comandos, comprobar el resultado de rotar vectores dirigidos en las direcciones x, y, -x, x+y, x-y segun angulos de pi, pi/2, y pi/4

Comprobar que rotMat(phi)*rotMat(-phi) es la matriz unidad para cualquier phi. ¿Por que?

Escribir un programa plotRotatedShape que cree n=5 puntos aleatorios en un cuadrado de lado unidad, centrado en el origen. Que despues los rote un angulo pi/4, los translade una unidad en la direccion x, y que dibuje el poligono cerrado que los une, antes y despues de rotarlos.

Pistas: generar los puntos en forma de matriz 2*(n+1), con el ultimo punto igual al primero, y usar rotMat para rotarlos todos a la vez; usar 'axis equal' para que las unidades en x e y sean iguales en la figura.

Dia 2

Escribir una funcion que calcule la posicion de un oscilador armonico amortiguado, con la siguiente interfaz:

```
function x = oscillator( x0, v0, omega, tau, t )
% Returns the position of a damped harmonic oscillator of the form
%   x(t) = ( A*cos(omega*t) + B*sin(omega*t) ) * exp(-t/tau)
% Input:
%   x0    : initial position (m)
%   v0    : initial velocity (m/s)
%   omega : angular frequency (rad/s)
%   tau   : damping time (s)
%   t     : time(s) at which x is to be found (s)
% Output:
%   x     : position of the oscillator at time(s) t (m)
% Usage:
%   x = oscillator( 1.5, 25., 3.5, 5., 10. )
```

Escribir un script plotOscillator.m que:
- Dibuje x(t) = oscillator(4, 15, 2.5, 3, t)
- Compruebe que la posicion y su derivada en t=0 coinciden con x0 y v0:

```
      oscillator(x0,v0,omega,tau,0) = x0
    ( oscillator(x0,v0,omega,tau,+dt/2) ...
     -oscillator(x0,v0,omega,tau,-dt/2) ) / dt = v0
```

Modificar la funcion anterior para que calcule tambien la velocidad en el instante t, con la siguiente interfaz:

```
function [x,v] = oscillator( x0, v0, omega, tau, t )
% Returns the position and velocity of a damped harmonic oscillator of the form
%   x(t) = ( A*cos(omega*t) + B*sin(omega*t) ) * exp(-t/tau)
% Input:
%   x0    : initial position (m)
%   v0    : initial velocity (m/s)
%   omega : angular frequency (rad/s)
%   tau   : damping time (s)
%   t     : time(s) at which x is to be found (s)
% Output:
%   x     : position of the oscillator at time(s) t (m)
```

```
%   v     : velocity of the oscillator at time(s) t (m/s)
% Usage:
%   [x,v] = oscillator( 1.5, 25., 3.5, 5., 10. )
```

Modificar plotOscillator para que dibuje tambien la velocidad y la energia mecanica del oscilador en funcion del tiempo.

Añadir una comprobacion de que la velocidad esta de acuerdo con la derivada de la posicion.

Escribir un script drawSpiral.m que dibuje una espiral

Escribir un script plotTwoGaussians.m que dibuje (en 3D) la suma de dos gausianas en 2D, de anchuras 0.2 y 0.3, y centradas en (0,0) y (1,0).
La funcion gausiana en 2D es z = C*exp(-((x-x0).^2+(y-y0).^2)/2/w^2), siendo (x0,y0) su centro, w su anchura, y C una constante que debera calcularse para que la integral de cada gausiana sea la unidad.
Sugerencia: usar sum para calcular la integral en una malla de puntos.

Escribir una funcion sphericalChargePotential que calcule el potencial electrostatico de una distribucion esferica de carga uniforme, con la interfaz

```
function V = sphericalChargePotential( Q, R, r )
% Calculates the electrostatic potential of a homogeneous spherical
% distribution of charge Q and radius R at a distance r, given by
%   V(r) = K*Q*(3*R^2-r^2)/(2*R^3)  inside  (r<R)
%        = K*Q/r                    outside (r>R)
% where K = 9e9 J*m^2/C^2 is Coulomb's constant
% Input:
%   Q  : total charge (C)
%   R  : radius of spherical charge distribution (m)
%   r  : distance to center of charge (m)
% Output:
%   V  : electrostatic potential (V)
% Usage:
%   V = sphericalChargePotential( -3.e-9, 0.2, 0.1 )
```

Escribir un script plotPotentialOfTwoCharges.m que dibuje, en el plano xy:
- La suma de dos distribuciones esfericas de carga uniforme, con las siguientes cargas totales, radios y posiciones, en unidades MKS:
  Q1=-2.e-9   R1=0.08   [x1,y1,z1]=[-1,0,0]
  Q2=+1.e-9   R2=0.05   [x2,y2,z2]=[+1,0,0]
- El potencial electrostatico de dichas cargas, con sphericalChargePotential.
- El campo electrico (usar gradient), en un plot de flechas (usar quiver), junto a las curvas de potencial constante (usar contour).
- La laplaciana (derivada segunda) del potencial (usar gradient del campo), comprobando que laplaciana(V) = d2V/dx2+d2V/dy2+d2V/dz2 = -4*pi*K*densidad

---