```
Ejercicios Tema 1
-----------------

----------------------------
Dia 1
----------------------------

Usar Matlab como calculadora:
Operaciones basicas y parentesis:
  2*2+15/3-2
  2*(2+15)/(3-2)
Potencias:
  5^2
  2*10^8
Variables:
  x=5*4
  y=2;  (probar con y sin ';' ¿Cual es la diferencia?)
  x/y
  x=x/5
Variables predeterminadas:
  ans
  2*pi
  3*i
  eps

Comandos utiles:
  format [short|long|compact]
  clear
  clc
  clf

Calcular la aceleracion de la gravedad g=G*M/r^2 para
la Tierra y la Luna (buscar G, M y r en internet)

Practicar funciones predeterminadas y numeros complejos:
  sqrt(-4)
  cos(pi)
  atan(1)*(180/pi)
  exp(2*pi*i)
  log(-1)

Matrices
  a = [1,3,5; 2,4,6]
  b = ones(3,2)
  c = eye(3)
  ab = a*b
  ba = b*a
  numel(a)
  size(a)
  size(a,1)
  size(a,2)
  size(b)
  size(ab)
  size(ba)

Vectores fila:
  x=[2 3 4 sqrt(25)]
  y=0.3:0.1:0.6
  z=linspace(0,100,6)
  x(1)
  x(3)
  y(2)
  numel(x)
  size(x)
  size(y)
  size(z)

Operaciones entre un escalar y un vector:
  x+1
  2*y
  z/10

Operaciones entre dos vectores:
  x+y
  x.*y  (observar el punto delante de *)
  x.^2

Vectores columna:
  a=[1,3,5]      (un vector fila)
  b=[2;4;6]      (un vector columna)
```

```
  c=a'           (otro vector columna, transpuesto de a)

Producto matricial de vectores:
  a*b     (¿entiendes el resultado?)
  b*a     (¿y este otro?)
  b'*a'   (¿y este?)

Dibujar una funcion:
  x=0:0.5:5;
  plot(x,x.^2)
  plot(x,x.^2,'o')
  plot(x,x.^2,'o',x,3*x,'+')
  xlabel('x values')
  ylabel('y(x) function')
  legend('x^2','3x')

Dibujar la trayectoria y(t)=y0+v0*t-0.5*g*t^2, con g=9.81m/s^2,
para las condiciones iniciales y0 = 15 m, v0 = 5 m/s
En caso de quedarse atascado, mirar la solucion: vertical.m

Hallar la maxima altura y en que tiempo ocurre (usar "zoom" con
una precision de dos digitos. Solucion: y=16.27m, t=0.51s)

Guardar la figura en un fichero firstVerticalTrajectory.pdf:
  print -dpdf firstVerticalTrajectory

------------------------------
Dia 2
------------------------------

Introduccion a archivos *.m

Escribir un programa plotTwoGaussians.m que dibuje
la funcion gausiana normalizada de anchura a,
y(x)=1/sqrt(2*pi*a^2)*exp(-x^2/2/a^2),
para a=1 y a=2, en dos curvas de la misma figura,
con leyendas 'a=1' y 'a=2', y etiquetas de 'x' e
'1/sqrt(2 \pi a^2) exp(-x^2/2a^2)' en los ejes x e y.
El titulo de la figura debe ser 'Two Gaussians'.

Escribir un programa plotTwoFunctions.m que dibuje
dos funciones lambda(x)=e^(-x^2) y Phi(x)=tanh(x)
en dos figuras distintas, con etiquetas de ejes 'x' y
'\lambda(x)' la primera y 'x' y '\Phi(x)' la segunda.

Escribir un programa plotSinXbyX.m que dibuje la funcion
sin(x)/x, y su derivada (x*cos(x)-sin(x))/x^2, en dos
subplots con etiquetas 'x','y=sin(x)/x' y 'x','dy(x)/dx'.
Primero poner los dos subplots uno al lado del otro
y luego uno encima del otro. Evitar el punto x=0.

Introduccion a las funciones.
   Encapsulamiento
   Argumentos de entrada
   Argumentos de salida

Escribir un programa programSetXandY.m que de valores a x e y.
Hacer 'clear', ejecutar el programa y comprobar los valores de x e y.

Repetir lo mismo pero con una funcion functionSetXandY.m

Escribir una funcion twoX.m que acepte un argumento 'x' y devuelva
el valor 2*x. Probarla para diversos valores de x.

Escribir una funcion twoXplusThreeY.m que acepte dos argumentos x e y,
devolviendo el valor 2*x+3*y.
Probarla para diversos valores de x e y.

Escribir una funcion sumAndProduct.m que acepte dos argumentos x e y,
devolviendo su suma y su producto.
Probarla para diversos valores de x e y.

Escribir una funcion con la interfaz
   function y=verticalTrajectory(y0,v0,t)
   % Given the initial vertical position y0 and velocity v0,
   % this function finds the trajectory y(t)=y0+v0*t-(g/2)*t^2
   % at the given times t, returning a vector of the same length as t

Escribir un programa plotVerticalTrajectory.m que defina un vector t,
llame a verticalTrajectory, y dibuje la trayectoria resultante y(t)
```

Escribir una funcion con la interfaz

```
   function [x,y]=parabolicTrajectory(x0,y0,v0x,v0y,t)
   % Given the initial position [x0,y0] and velocity [v0x,v0y],
   % this function finds the trajectory x(t)=x0+v0x*t, y(t)=y0+v0y*t-(g/2)*t^2
   % at the given times t, returning two vectors of the same length as t
```

Escribir un programa plotParabolicTrajectory.m que defina un vector t, llame a parabolicTrajectory, y dibuje la trayectoria resultante y(x).

Dia 3

Estructuras de control basicas:
   Bucles 'for'
   Condiciones 'if-else'

Escribir una funcion derivative.m que devuelva la funcion derivada de una funcion y(x) definida en puntos arbitrarios x, con la interfaz

```
function dydx=derivative(x,y)
% Given a function y as a vector of values at points x, it returns its
% derivative at the same points. Uses a simple finite-difference approximation:
%   dydx(i) = (y(i+1)-y(i-1)) / (x(i+1)-x(i-1))   for 1<i<n
%   dydx(1) = (y(2)-y(1)) / (x(2)-x(1))
%   dydx(n) = (y(n)-y(n-1)) / (x(n)-x(n-1))
% Input:
%   x    : vector of x points
%   y    : vector of y values at given x points
% Output:
%   dydx : vector of dy/dx at given x points
% Usage example:
%    x = 0:pi/100:2*pi
%    y = sin(x)
%    z = derivative(x,y)
```

Escribir un programa testDerivative.m que llame a derivative con una funcion y(x)=sin(x) y compare la derivada resultante con el resultado exacto.

Escribir una funcion integral.m que calcule la integral de una funcion y(x), definida en puntos arbitrarios x, con la interfaz

```
function z=integral(x,y)
% Given a function y as a vector of values at points x, returns its
% definite integral between the first point and each of the other points.
% Uses a linear interpolation approximation for y(x) between points.
% Input:
%   x    : vector of x points
%   y    : vector of y values at given x points
% Output:
%   z    : vector of Int_xmin^x y(x') dx', at given x points
% Usage example:
%    x = 0:pi/100:2*pi
%    y = sin(x)
%    inty = integral(x,y)
```

Escribir un programa testIntegral.m que llame a integral con una funcion y(x)=cos(x) y compare la integral resultante con el resultado exacto.

Escribir un programa testParabolicTrajectory.m que calcule una trayectoria llamando a parabolicTrajectory y luego calcule la velocidad llamando a derivative para las componentes x e y. Dibujar el resultado, junto a la velocidad analitica vx(t)=v0x, vy(t)=v0y-g*t en la figura 1. Volver a llamar a derivative para calcular la aceleracion, y dibujarla en la figura 2 junto a la aceleracion conocida ax(t)=0, ay(t)=-g.
