# 1 Enunciados: Leer ficheros CSV

## 1.1 Ejercicio 1

El fichero *'slopes.csv'* contiene la abcisa *s* y la pendiente *p* de puntos cada cinco metros de un tramo de la carretera M-607 de Madrid, en formato CSV (Valores en cada fila separados por una coma). Se pide desarrollar una función que lea el fichero, almacenando los valores leídos en sendos vectores *s* y *p*. La función calculará e imprimirá en pantalla: (1) el número de puntos leídos, (2) la longitud del tramo y la pendiente media del tramo. En el caso de que la función no pudiera leer el fichero, mostrará un mensaje en pantalla informando del error, y terminará la ejecución.

**Nota:** *El fichero de datos slopes.csv, necesario para la realización del ejercicio, se puede descargar desde la plataforma moodle de la asignatura.*

## 1.2 Ejercicio 2

Desarrollar una función que lea el fichero *'slopes.csv'* y dibuje el gráfico de pendientes.

# 2 Soluciones

## 2.1 Ejercicio 1: Solución

Es frecuente encontrarse ficheros en formato CSV, esto es, con ficheros que tienen los valores dentro de cada línea separados por coma o punto y coma u otro caracter separador, distinto del espacio. Una manera eficaz de enfrentarse a estos ficheros es abrirlos con la hoja de cálculo y desde ahí hacer una exportarción a formato CSV pero utilizando el espacio como separador. Una vez grabado el fichero con espacios como separador de elementos en la fila, podremos procesarlo con las instrucciones de Matlab-Octave que están dentro del temario. Otra manera de acometer el problema sería leer cada línea como línea de texto, hacer un replace, *strrep()*, cambiando la coma por un espacio, y luego leer los datos de la cadena resultante con *sscanf()*. Esa solución es la que se muestra a continuación.

```
function [s, p] = readSlopes(filename)
   % Lee un fichero CSV con valores abcisa pendiente separados por coma
   % Devuelve un vector columna s con las abcisas y un vector columna p
   % con las pendientes

   s = [];
   p = [];

   fprintf('readSlopes() : Reading file %s ...',filename)

   fid = fopen(filename, 'r');
   if(fid <3)
      fprintf(' ERROR: can''t read file\n', filename)
      return;
   end

   fprintf(' OK, result %d\n', fid)

   count =1;
   s = [];
   p = [];
   pacum = 0.0;
   while(~feof(fid))
      cad = fgetl(fid);
      cad = strrep(cad,',',' ');
```

```
        v = sscanf(cad, '%f');
        s(count) = v(1);
        p(count) = v(2);
        pacum = pacum + p(count);
        count++;
    end

    fclose(fid);

    numpuntos = length(s);
    fprintf('Leidos %d puntos\n', numpuntos)

    ltramo = s(end) - s(1);
    fprintf('Longitud del tramo= %12.2f\n', ltramo)

    pmedia = pacum / numpuntos;
    fprintf('Pendiente media= %f\n', pmedia)

end
```

En el caso de no querer utilizar la función *sscanf()*, se podría ir leyendo cada línea del fichero, hacer el replace de la coma por el espacio, e ir escribiendo las líneas con espacio en un nuevo fichero. Tras procesar todo el fichero original, podríamos hacer una segunda lectura del nuevo fichero generado con las funciones que entran en el temario. Esa solución se muestra a continuación.

```
function [s, p] = readSlopes2(filename)
% Lee un fichero CSV con valores abcisa pendiente
    s = [];
    p = [];

    fprintf('readSlopes() : Reading file %s ...',filename)

    fin = fopen(filename, 'r');
    fout = fopen('temp.tmp', 'w');
    if(fin < 3 | fout < 3)
        fprintf(' ERROR: can''t open files\n')
        return;
    end

    fprintf(' OK, result %d\n', fin)

    % Copiar a fichero temporal cambiando comas por espacios
    while(~feof(fin))
        cad = fgetl(fin);
        cad = strrep(cad,',',' ');
        if(length(cad) > 0)
            fprintf(fout, '%s\n', cad);
        end
    end

    fclose(fin);
    fclose(fout);

    fid = fopen('temp.tmp', 'r');
    if(fid < 3)
        fprintf(' ERROR: can''t open temp file\n')
        return;
```

```
    end

    s = [];
    p = [];
    pacum = 0.0;
    count = 0;
    while(~feof(fid))
        count++;
        x = fscanf(fid, '%f %f', [1,2]);
        if(size(x,2) == 2)
            s(count) = x(1);
            p(count) = x(2);
            pacum += p(count);
        end
    end

    numpuntos = length(s);
    fprintf('Leidos %d puntos\n', numpuntos)

    ltramo = s(end) - s(1);
    fprintf('Longitud del tramo= %12.2f\n', ltramo)

    pmedia = pacum / numpuntos;
    fprintf('Pendiente media= %f\n', pmedia)
end
```

## 2.2 Ejercicio 2: Solución

Se puede utilizar la misma función del ejercicio anterior para la lectura del fichero:

```
function plotSlopes(filename)
% Dibuja el grafico de pendientes de un fichero CSV de abcisas, pendientes
    [s, p] = readSlopes(filename);
    if(length(s) > 0 & length(s) == length(p))
        plot(s,p);
    end
end
```