

# **Seguridad, Privacidad y aspectos legales y éticos**

## **Práctica 2 - Sistema de verificación de firma online con Matlab**

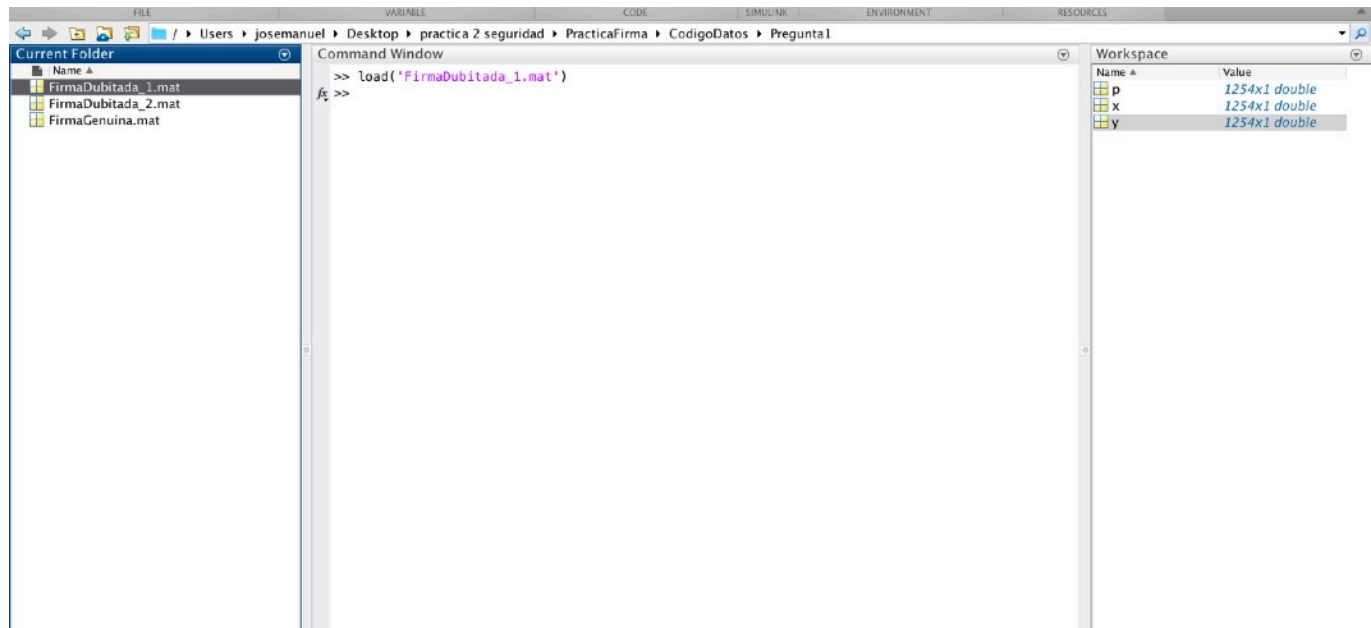
José Manuel Bustos Muñoz

## **Índice**

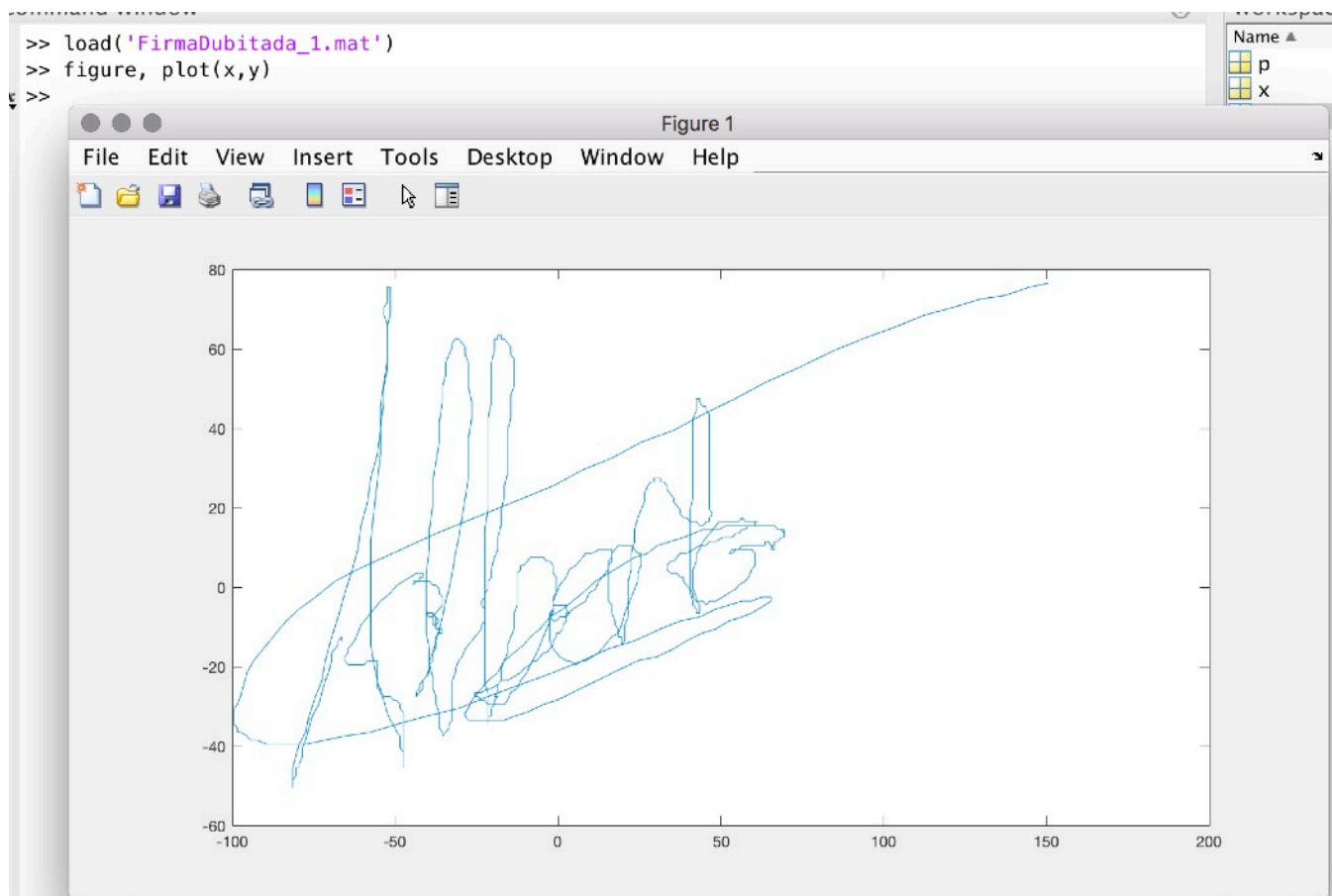
1. Analizar gráficamente y visualmente varias firmas y sus parámetros.
2. Crear funciones que extraigan características y representar las distribuciones obtenidas.
3. Calcular el score entre las firmas registradas de un usuario y la firma de test con la distancia euclídea.

## 1. Analizar gráficamente y visualmente varias firmas y sus parámetros.

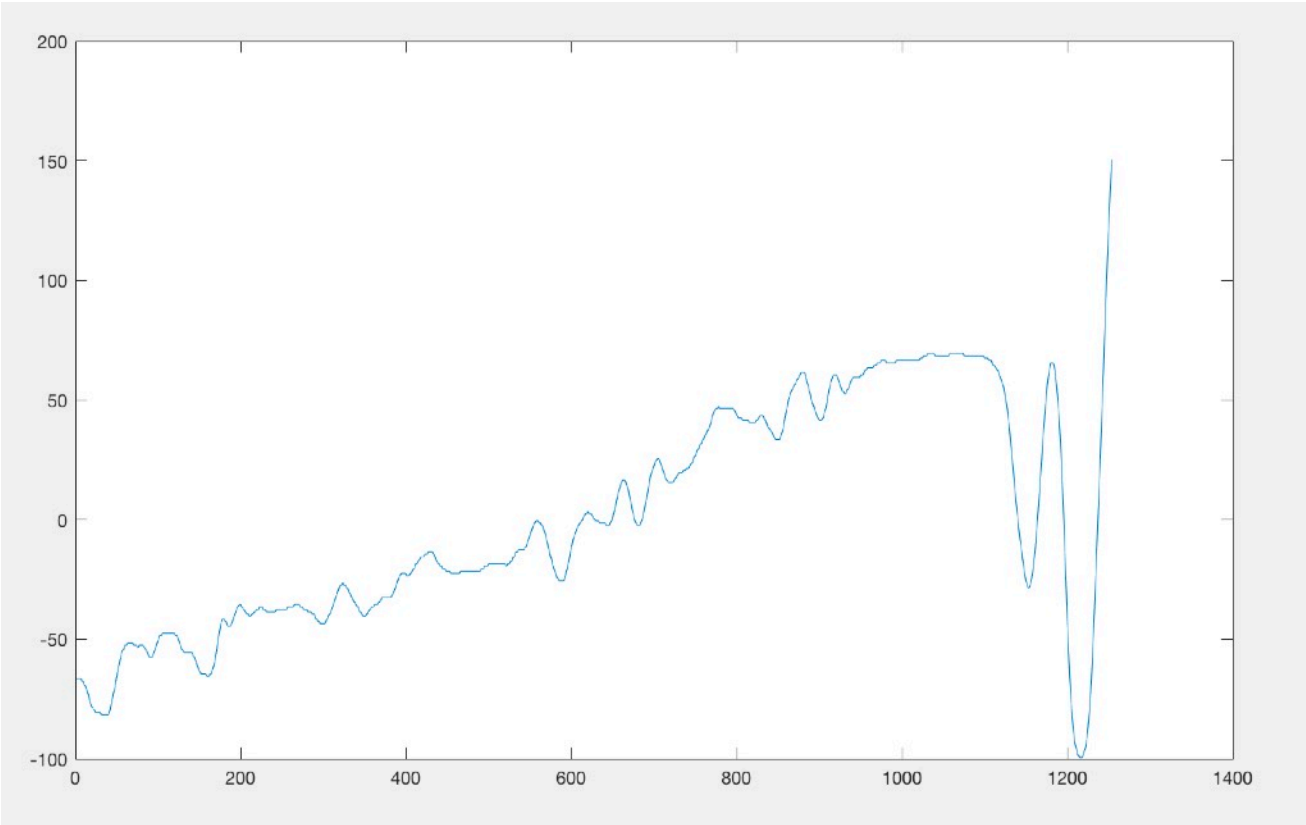
Abrimos la aplicación Matlab, cargamos las 3 firmas para el primero ejercicio, y cargamos una de ellas y vemos como en el menú de la derecha se cargan sus características 'x', 'y', y 'p'.



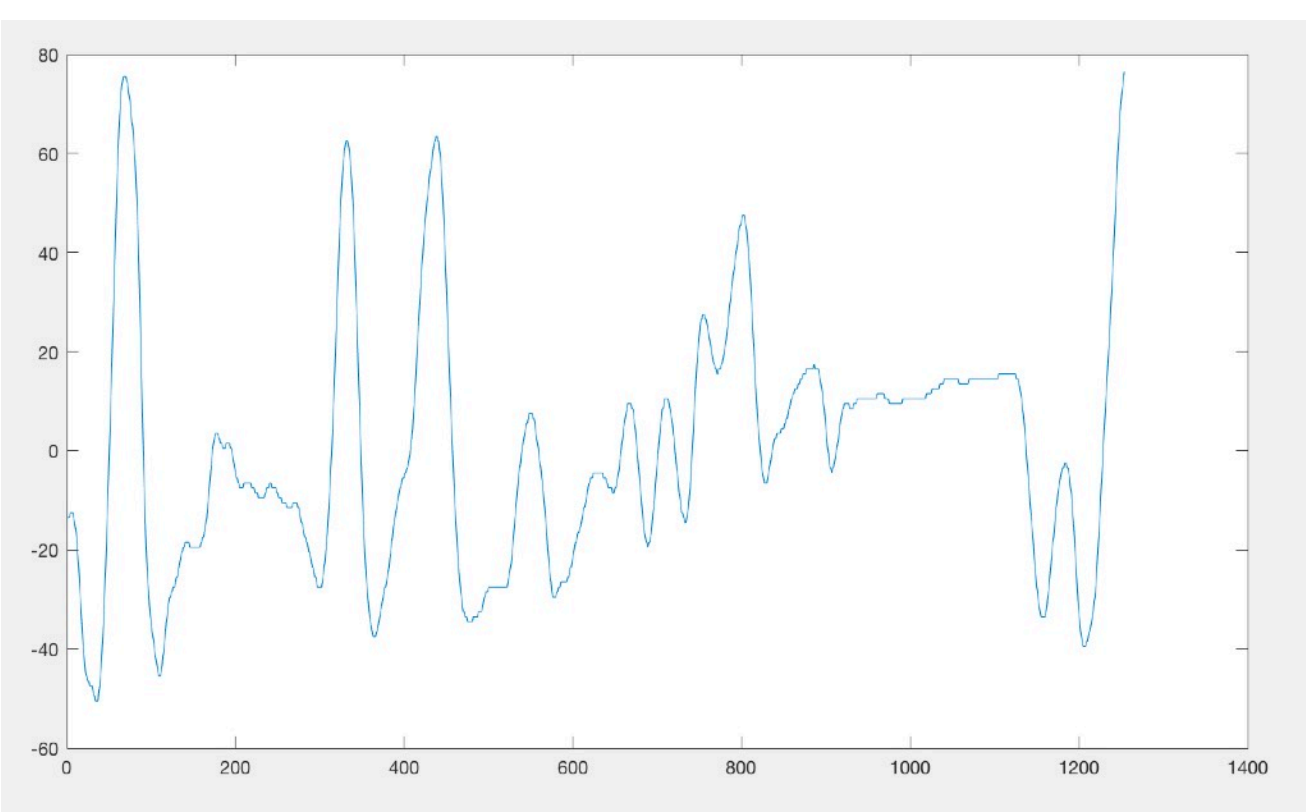
Para pintar cada una de las firmas cargadas, lo hacemos con “figure, plot(x,y)” y salta la ventana con la firma representada por sus ejes x e y. Luego para ver representada cada una de las variables de cada firma, usaremos la función plot() con x, y, p.



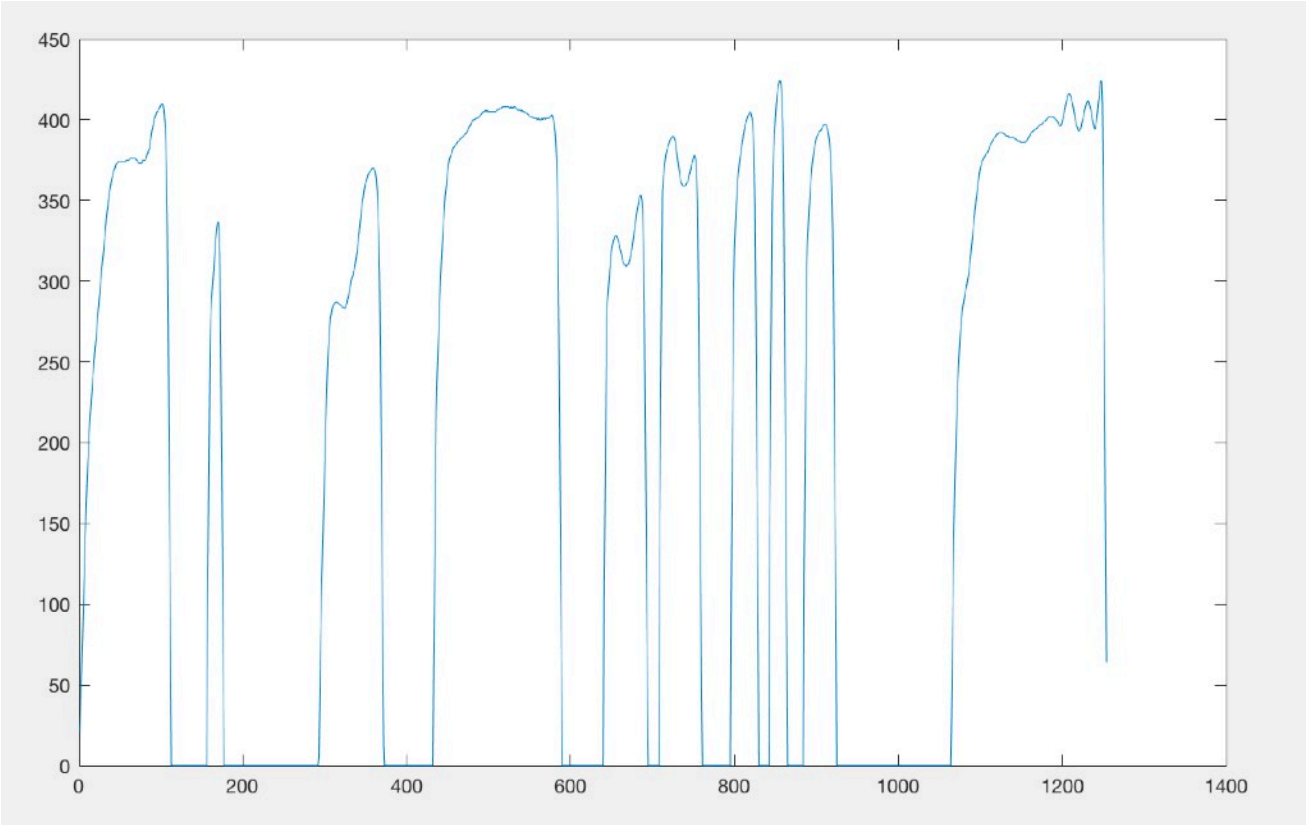
Representación x - Firma dubitada 1



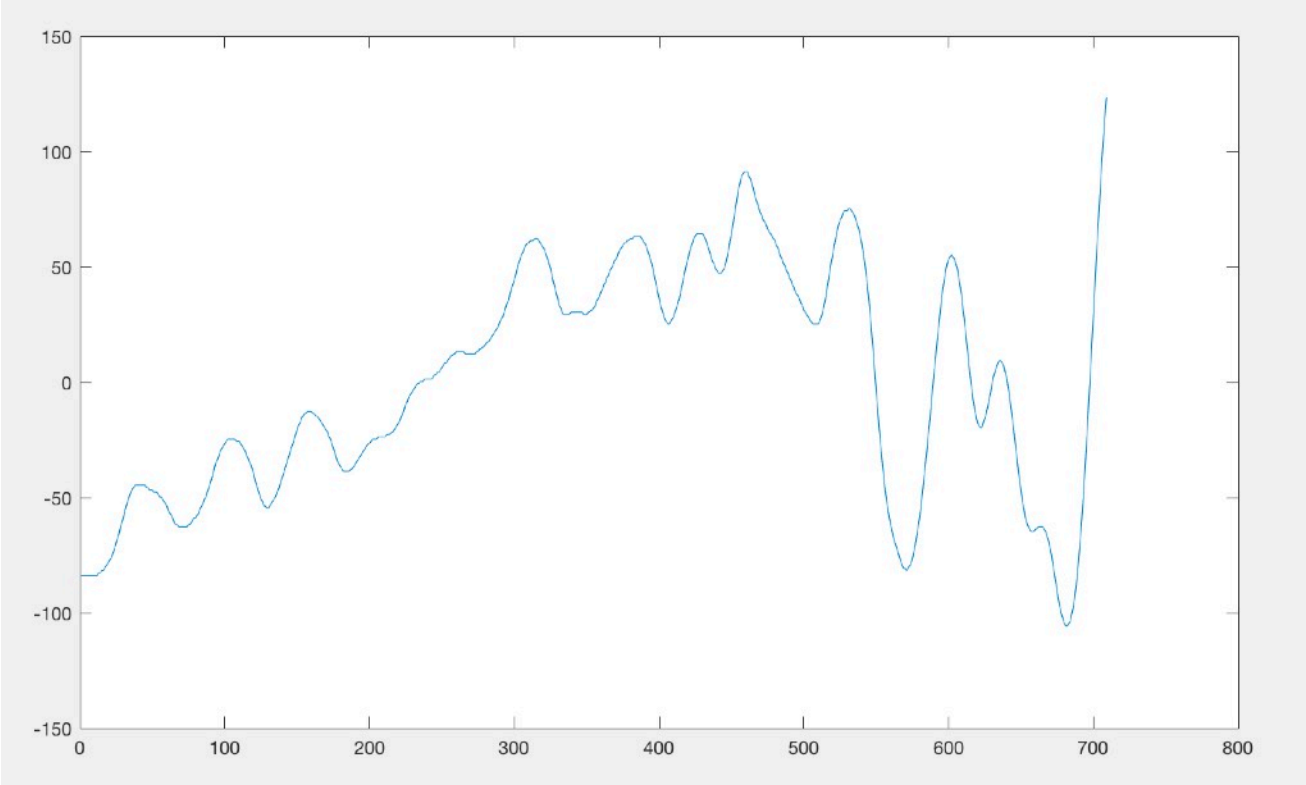
Representación y - Firma dubitada 1



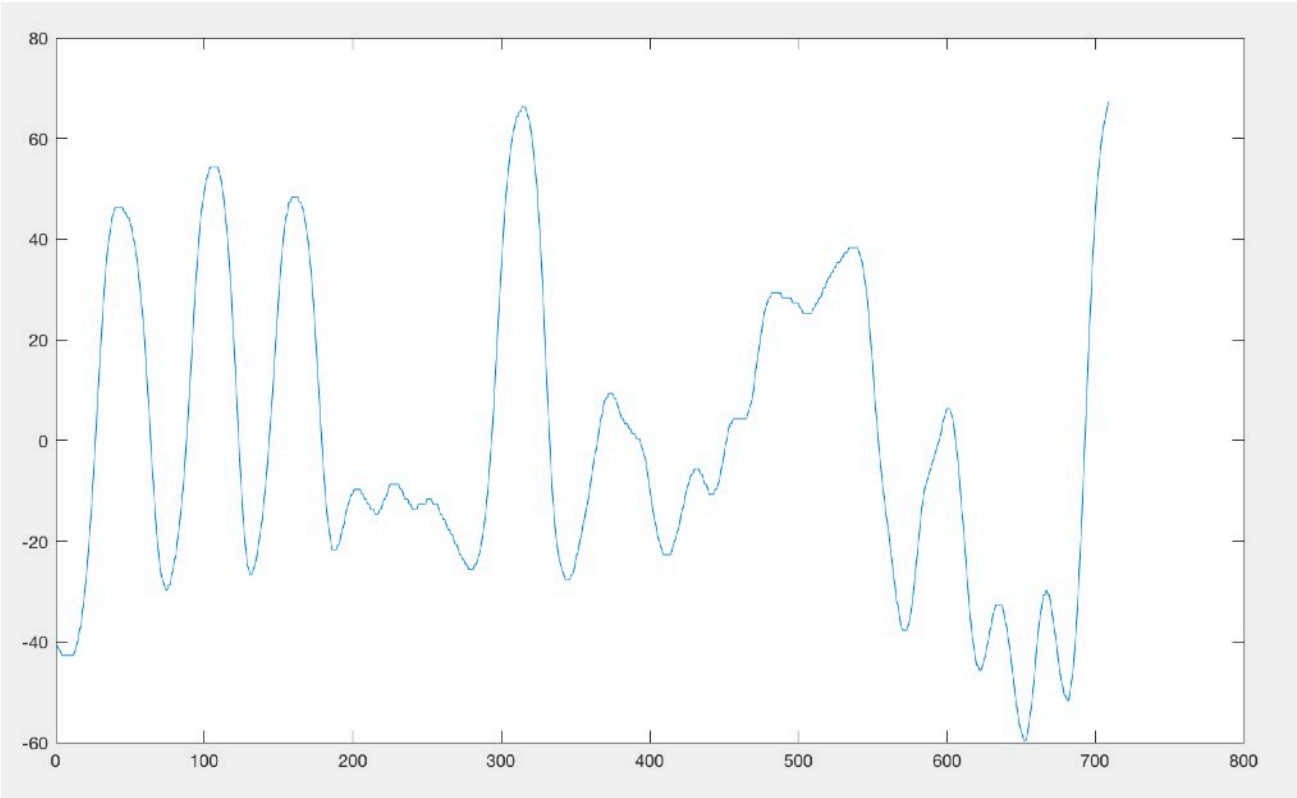
Representación p - Firma dubitada 1



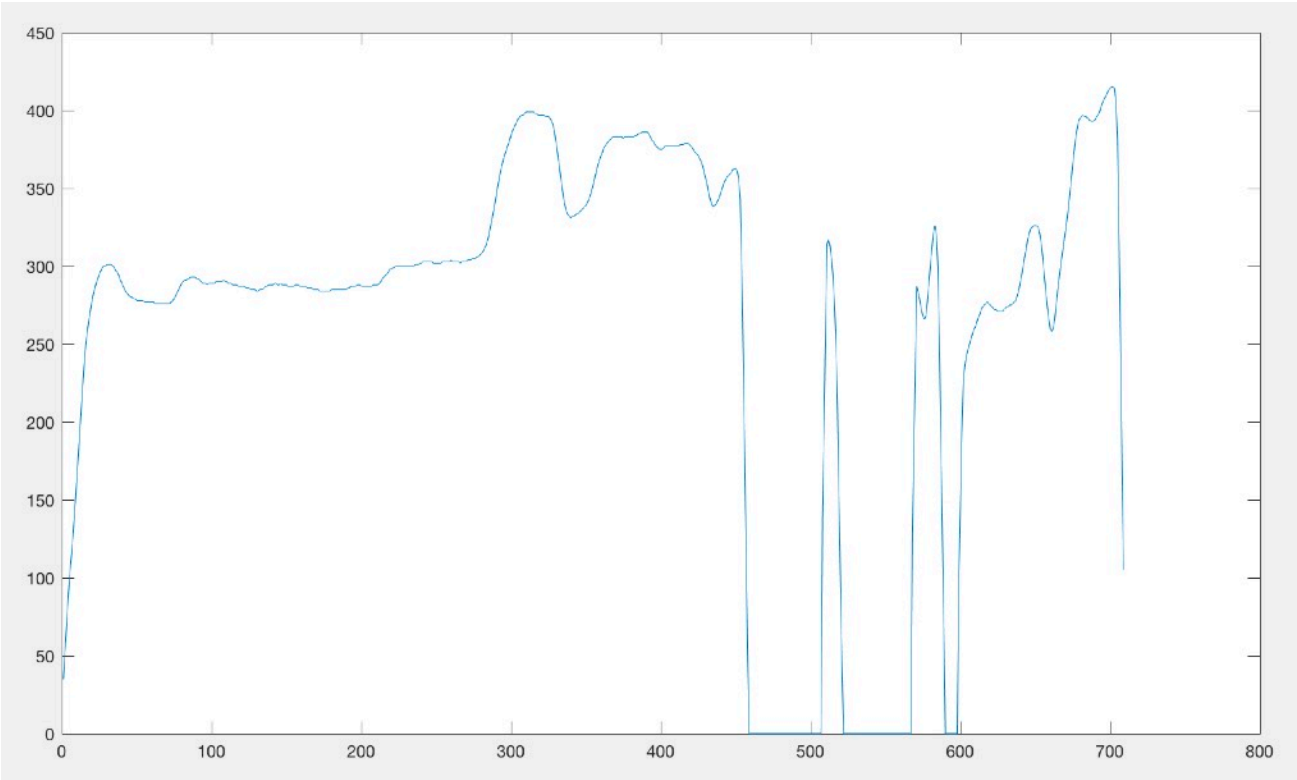
Representación x - Firma dubitada 2



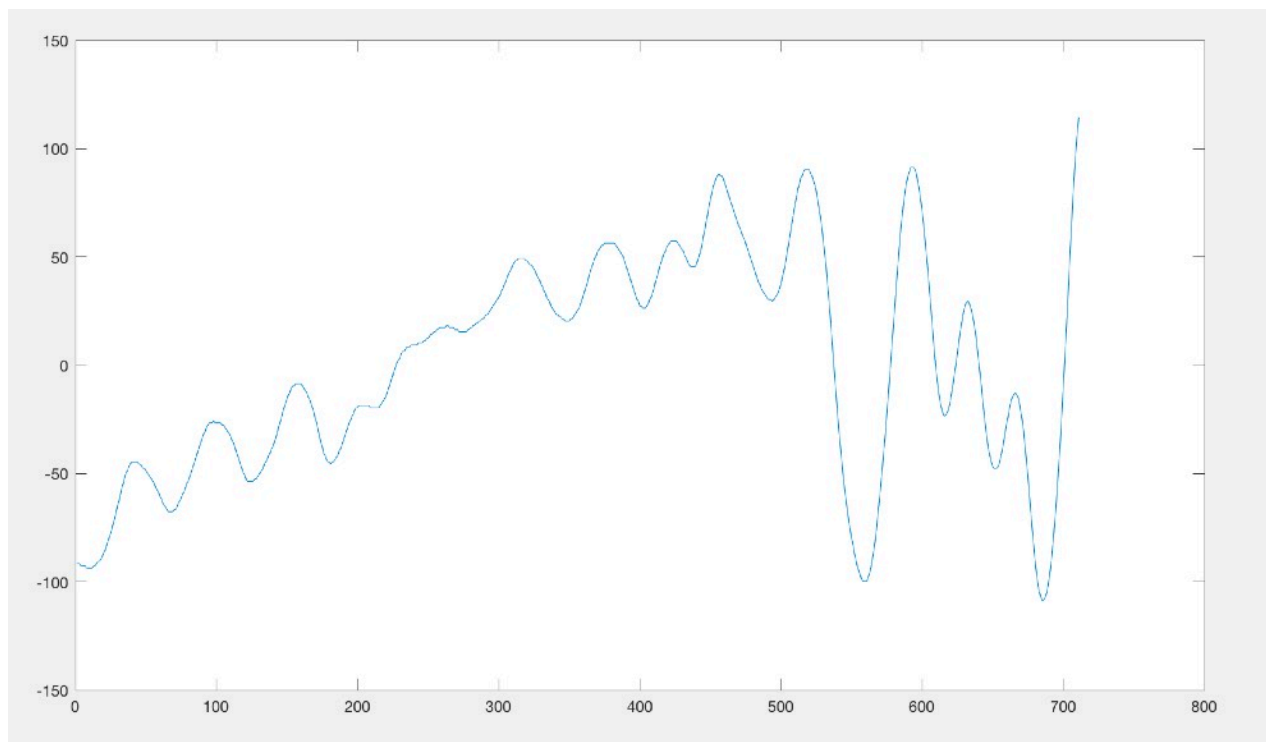
Representación y - Firma dubitada 2



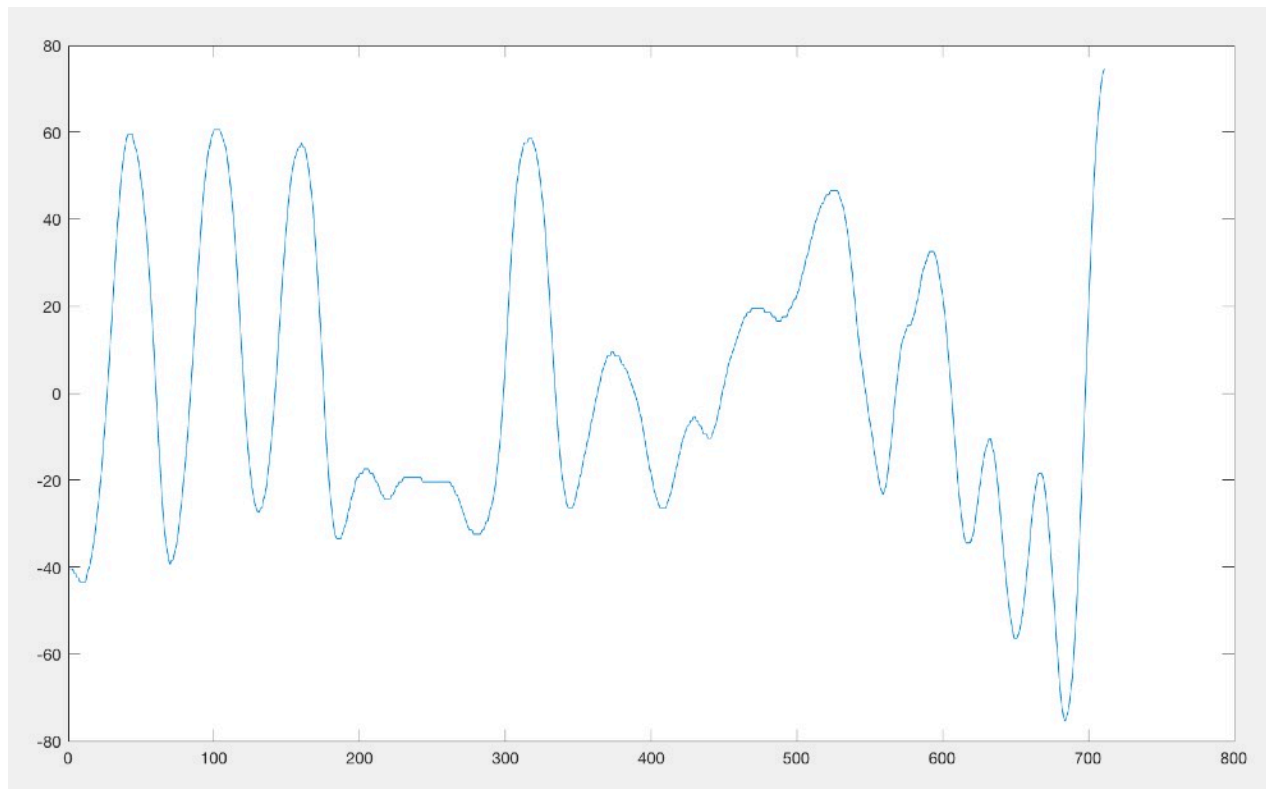
Representación p - Firma dubitada 2



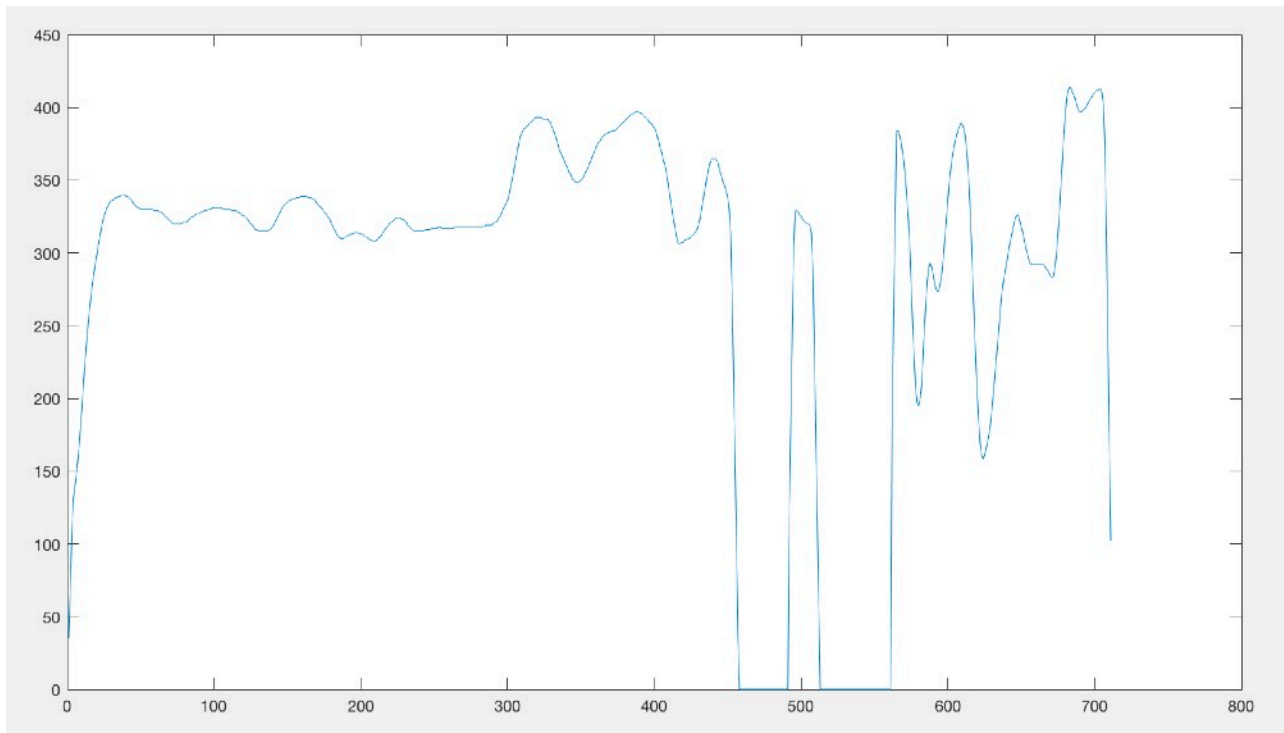
Representación x - Firma genuina



Representación y - Firma genuina



## Representación p - Firma genuina



Vemos los valores de cada vector: x, y, p. La longitud de los vectores será el número de muestras de cada una de las firmas.

Firma genuina:

Name	Value
p	711x1 double
x	711x1 double
y	711x1 double

Firma dubitada 1:

Name	Value
p	1254x1 double
x	1254x1 double
y	1254x1 double

Firma dubitada 2:

Name	Value
p	709x1 double
x	709x1 double
y	709x1 double



Suponemos que se han cogido 200 muestras por segundo, entonces para ver la duración en segundos de cada firma dividimos la longitud de uno de sus vectores por 200 y obtenemos las duraciones:

Firma genuina - 3,5550 segundos.  
 Firma dubitada 1 - 6,27 segundos.  
 Firma dubitada 2 - 3,5450 segundos.

```
>> duracion_dubitada2 = length(x)/200

duracion_dubitada2 =

    3.5450

>> load('FirmaDubitada_1.mat')
>> duracion_dubitada1 = length(x)/200

duracion_dubitada1 =

    6.2700

>> load('FirmaGenuina.mat')
>> duracion_genuina = length(x)/200

duracion_genuina =

    3.5550
```

Ahora calculamos para cada firma el número de levantamientos y el número de trazos escritos. Para calcular el nº de levantamientos vamos a ver en el vector 'p' de cada firma el número de valores que son iguales a 0, que sería cuando no hay presión.

Se crea una variable igualada a 0 inicialmente, y posteriormente con un bucle for recorreremos el vector 'p' de cada firma buscando el número de valores igual a 0, que serán los que identifican los 'pen ups' o número de levantamientos.

Para la firma genuina se obtienen: 83 pen ups, 628 pen downs.

```
>> for j=1:length(p)
if (p(j)==0)
pen_ups_genuina = pen_ups_genuina+1
end
end
```

j	711
p	711x1 double
pen_ups_genuina	83
x	711x1 double
y	711x1 double

Para la firma dubitada 1 se obtienen: 493 pen ups, 761 pen downs.

```
>> for z=1:length(p)
if (p(z)==0)
pen_ups_dubitada = pen_ups_dubitada+1
end
end
```

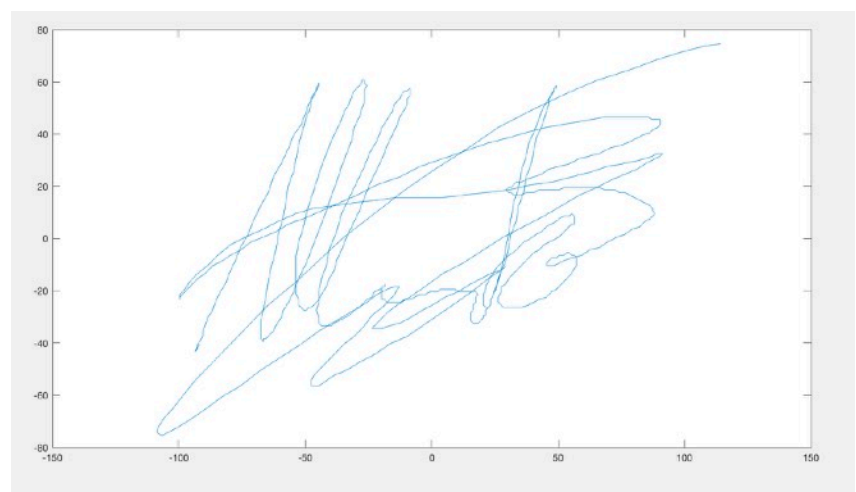
p	1254x1 double
pen_ups_dubitada	493
pen_ups_genuina	83
x	1254x1 double
y	1254x1 double
z	1254

Para la firma dubitada 2 se obtienen: 104 pen ups, 605 pen downs.

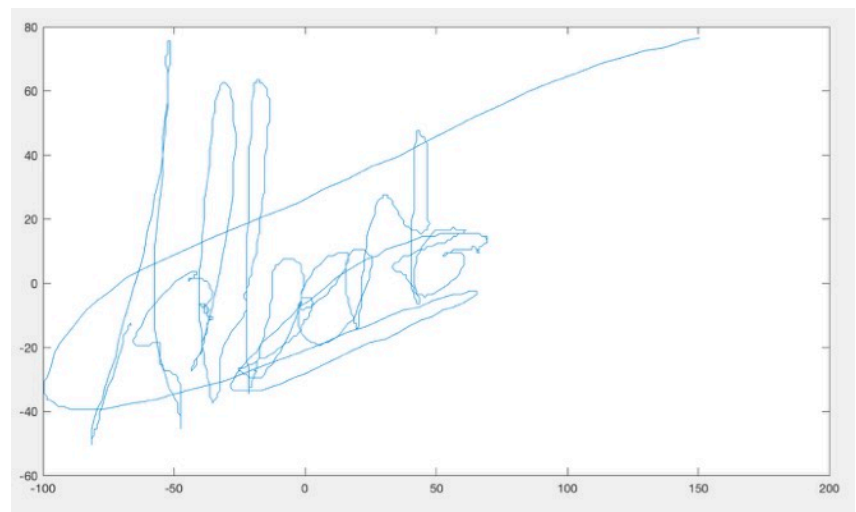
```
>> for w=1:length(p)
if (p(w)==0)
pen_ups_dubitada2 = pen_ups_dubitada2+1
end
end
```

j	711
p	709x1 double
pen_ups_dubitada	493
pen_ups_dubitada2	104
pen_ups_genuina	83
w	709
v	709x1 double

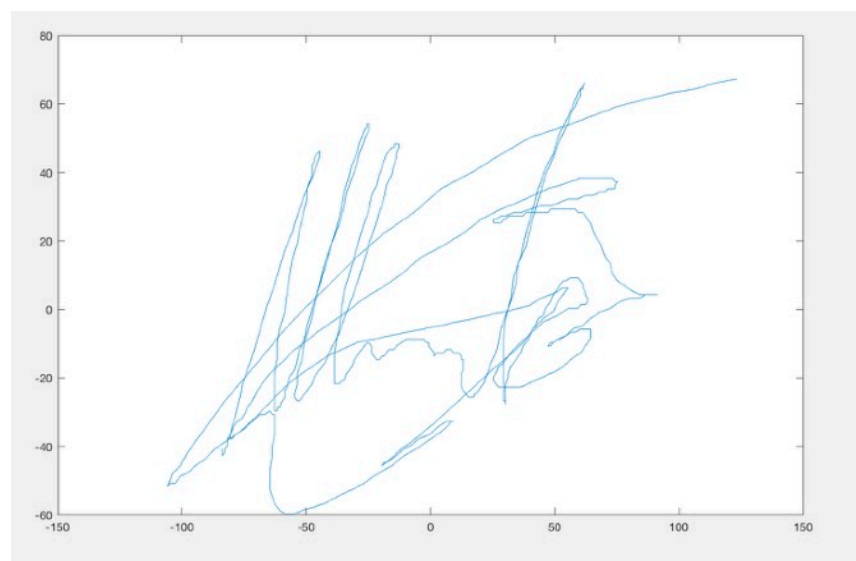
Ahora presentamos gráficamente las 3 firmas:  
Firma genuina



Firma dubitada 1



Firma dubitada 2



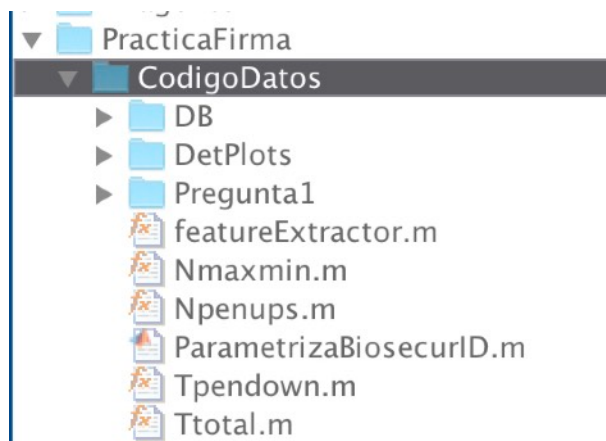
Si se tienen en cuenta los valores de duración de cada firma, la longitud de las firmas, y los valores de pen ups y pen downs de cada una, se aprecia como la firma dubitada 2 tiene valores muy similares a la firma genuina, mientras la firma dubitada 1 tiene valores muy distintos a las otras dos.

Además, si se comparan las representaciones de las 3 firmas, y también las representaciones de los ejes x e y, **la firma dubitada 1 en apariencia es muy diferente a la firma genuina, y da la impresión de ser una firma falsificada. La firma dubitada 2 si podría ser una firma verídica si se tienen en cuenta los valores calculados y las representaciones gráficas.**

## 2. Crear funciones que extraigan características y representar las distribuciones obtenidas.

Creamos en la misma ruta que están “ParametrizaBiosecurID.m” y “featureExtractor.m” los ficheros para las 4 funciones que vamos a generar y que utilizaremos en el ejercicio. Las 4 funciones son:

- Nmaxmin: obtiene el número de máximos y mínimos que tenga la característica pasada.
- Npenups: obtiene el número de veces que se levanta el lapiz.
- Tpendown: obtiene el tiempo en el cual hay presión.
- Ttotal: el tiempo.



Función Ttotal: la longitud del vector de la característica pasada, dividido por 200 que son las muestras recogidas por segundo. Así se obtienen los segundos que duró la firma.

```
Npenups.m x Tpendown.m x Ttotal.m x featu
function T = Ttotal(x)
    T = length(x)/200;
end
```

Prueba de la función Ttotal: le pasamos un vector con las muestras de un eje x, y se obtiene 3,5450 como el tiempo de la firma representada por esa característica.

```
>> tiempo_total = Ttotal(x)

tiempo_total =

    3.5450
```

Función Npenups: con un bucle for recorremos el vector de la característica “p” y se saca el número de veces que la muestra toma el valor 0, ya que es cuando se levanta el lápiz y no se está ejerciendo presión.

```
Npenups.m  Tpendown.m  Ttotal.m  f
function Npu = Npenups(p)
    Npu = 0
    for i = 1:length(p)
        if (p(i)==0)
            Npu = Npu+1
        end
    end
end
```

Función Tpendown: primero calculamos el tiempo total como vimos anteriormente de dividir la longitud del vector por 200. Luego recorremos el vector ‘p’ para encontrar el número de muestras que toman valor distinto a 0, que serán las muestras donde se está firmando al haber presión.

Una vez tenemos este número, lo dividimos entre 200 y se obtendría el tiempo en el cual se estaba ejerciendo presión.

```
Npenups.m  Tpendown.m  Ttotal.m  fe
function Tpd = Tpendown(p)
    total = length(p)/200;
    a = 0
    for i = 1:length(p)
        if (p(i)~=0)
            a = a+1
        end
    end
    npd = a;
    Tpd = npd/200;
end
```

Función Nmaxmin: los máximos se obtienen con el método findpeaks(), y su longitud será el número de máximos del vector pasado al método anterior. Luego, para sacar los mínimos se calcularía el vector inverso y sobre él podría hacerse como con los máximos.

```
Nmaxmin.m  x  +
function Nmax = Nmaxmin(x)
-
-     maximos = findpeaks(x);
-     nummaximos = length(maximos);
-     DataInv = 1.01*max(x) - x;
-     [Minima,MinIdx] = findpeaks(DataInv);
-     minimos = x(MinIdx);
-     numminimos = length(minimos);
-
-     Nmax = nummaximos+numminimos;
-
- end
```

Prueba de la función Nmaxmin: para el vector x pasado a la función nos dice que el número de máximos y mínimos serían 27.

```
>> n = Nmaxmin(x)

n =

    27

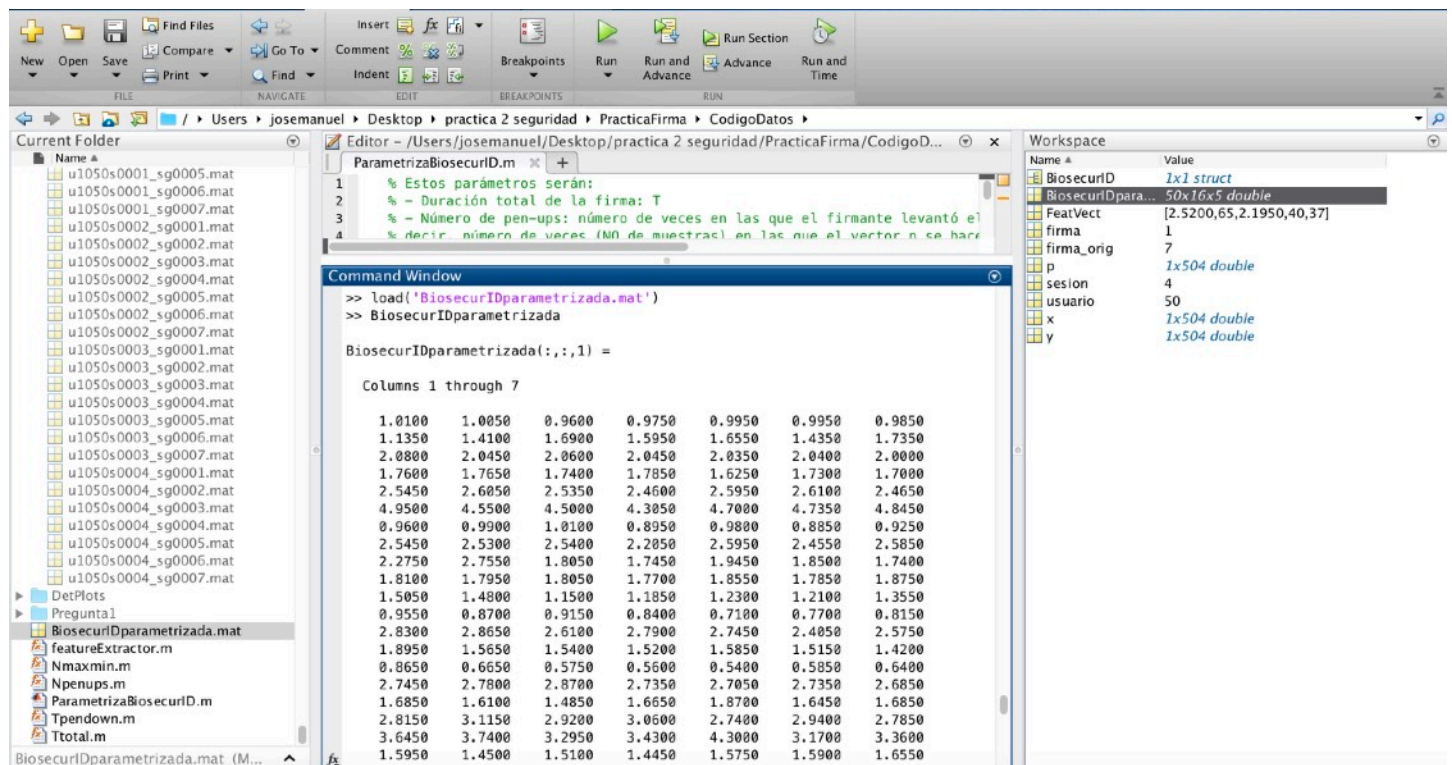
>>
```

Probamos la función “featureExtractor” que utiliza todas las funciones anteriores. Le pasamos los vectores x, y, p de una de las firmas, y obtenemos un vector que se puede ver en la imagen que contiene los siguientes valores: 3.5550, 83, 3.1400, 27, 27.

```
>> load('/Users/josemanuel/Desktop/practica 2 seguridad/Practica 2.mat')
>> vector = featureExtractor(x,y,p)
..
```



Al correr el programa “BiosecurIDparametrizada.mat” se genera la matriz “BiosecurIDparametrizada”.

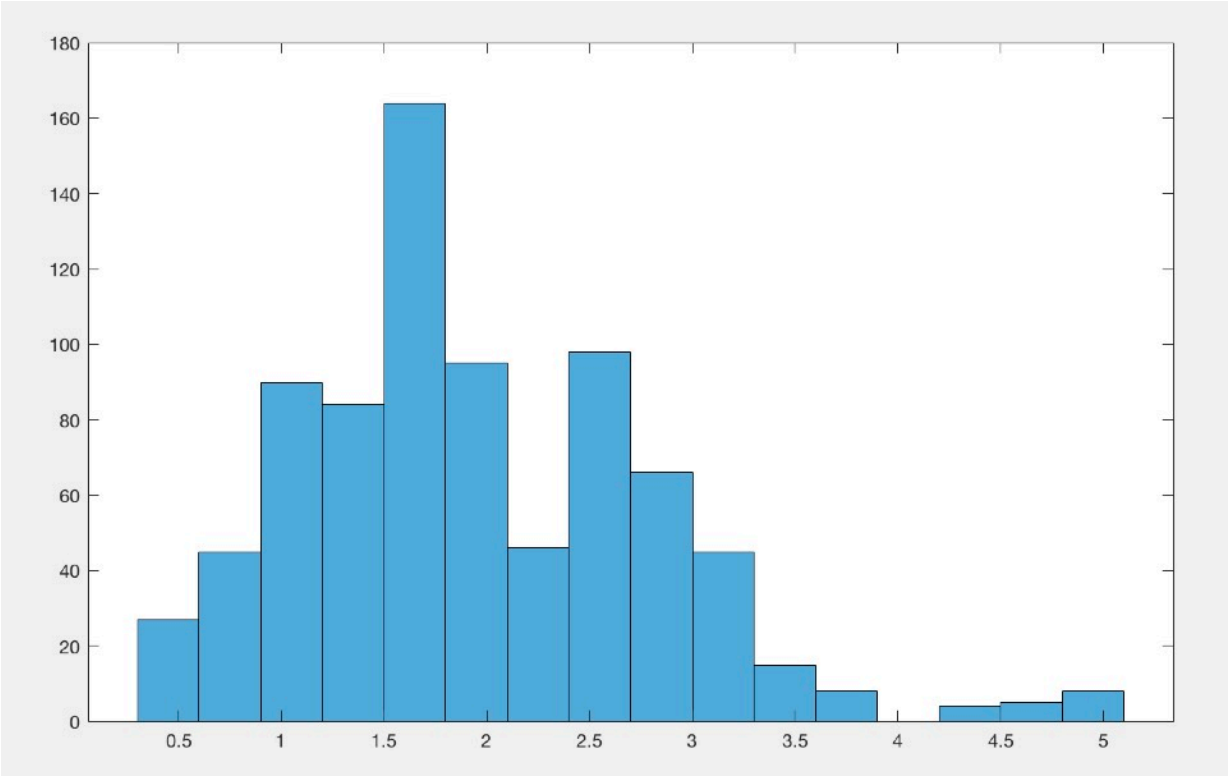


Para pintar los histogramas de las cinco características del fondo de la matriz, podemos usar la función “hist” o la función “histogram” como en las siguientes imágenes. Nos decantamos por “histogram” que tiene una mejor visualización.

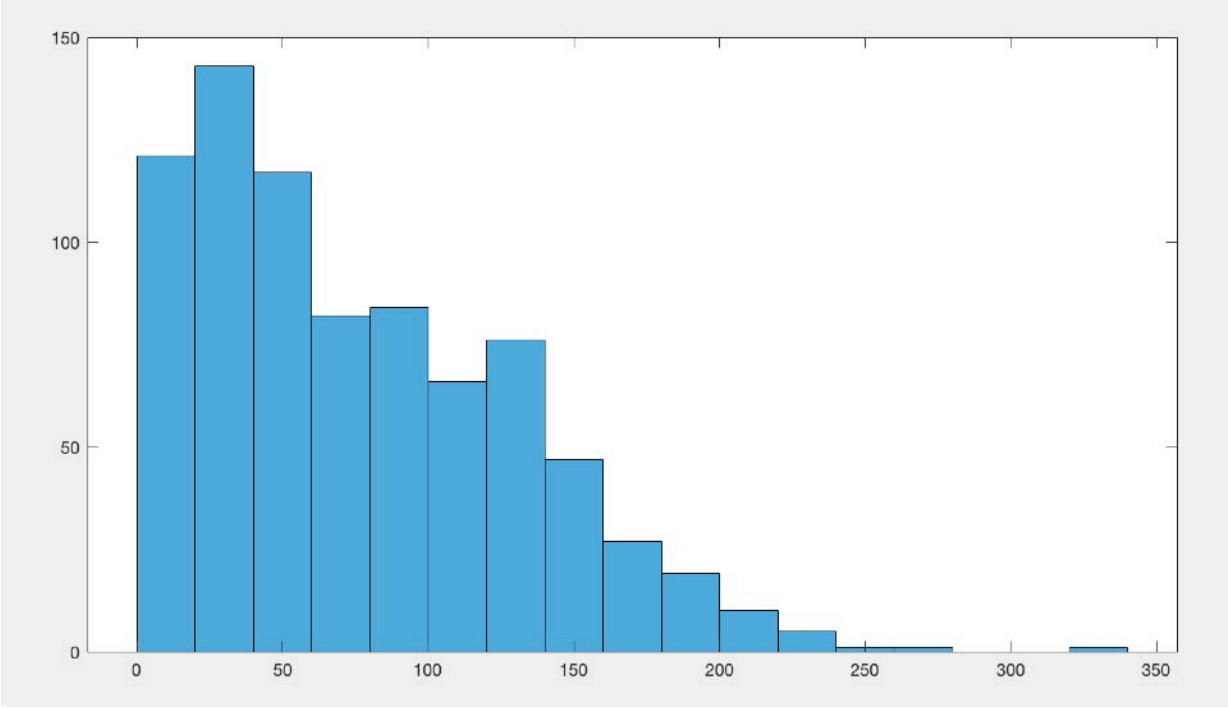
```
>> figure, hist(BiosecurIDparametrizada(:, :, 1))
>> figure, hist(BiosecurIDparametrizada(:, :, 2))
>> figure, hist(BiosecurIDparametrizada(:, :, 3))
>> figure, hist(BiosecurIDparametrizada(:, :, 4))
>> figure, hist(BiosecurIDparametrizada(:, :, 5))
```

```
>> figure, histogram(BiosecurIDparametrizada(:, :, 1))
>> figure, histogram(BiosecurIDparametrizada(:, :, 2))
>> figure, histogram(BiosecurIDparametrizada(:, :, 3))
>> figure, histogram(BiosecurIDparametrizada(:, :, 4))
>> figure, histogram(BiosecurIDparametrizada(:, :, 5))
```

Duración total T:

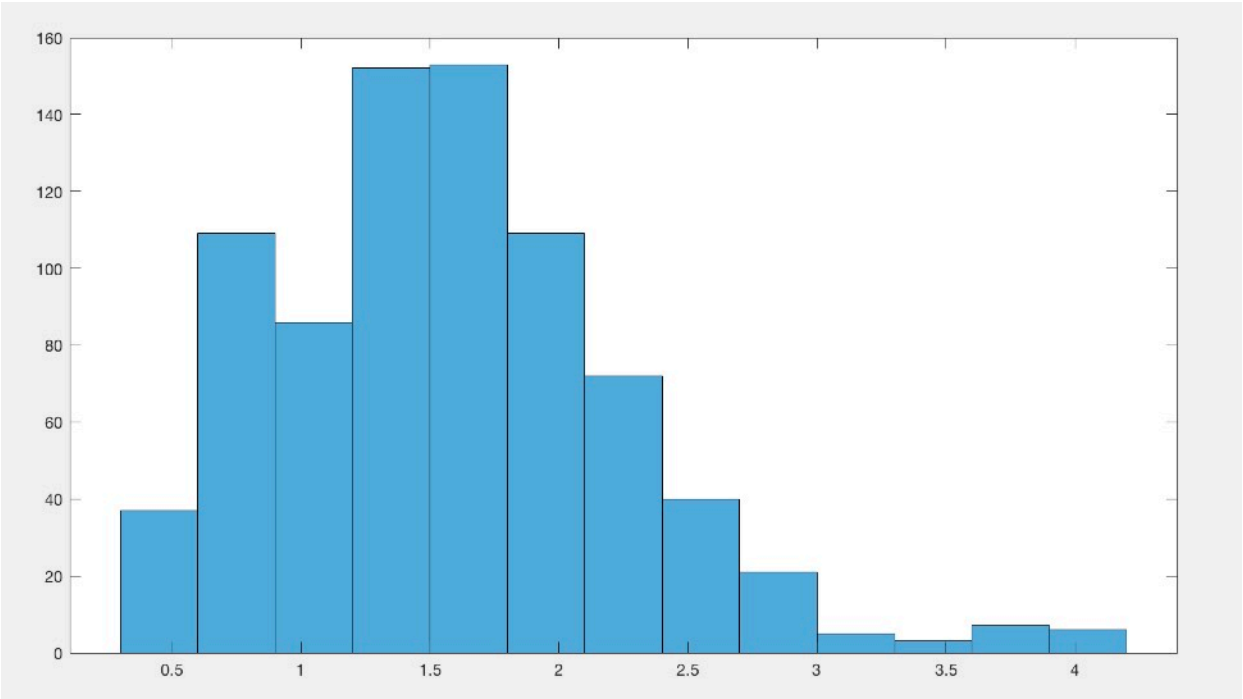


Número de pen-ups:

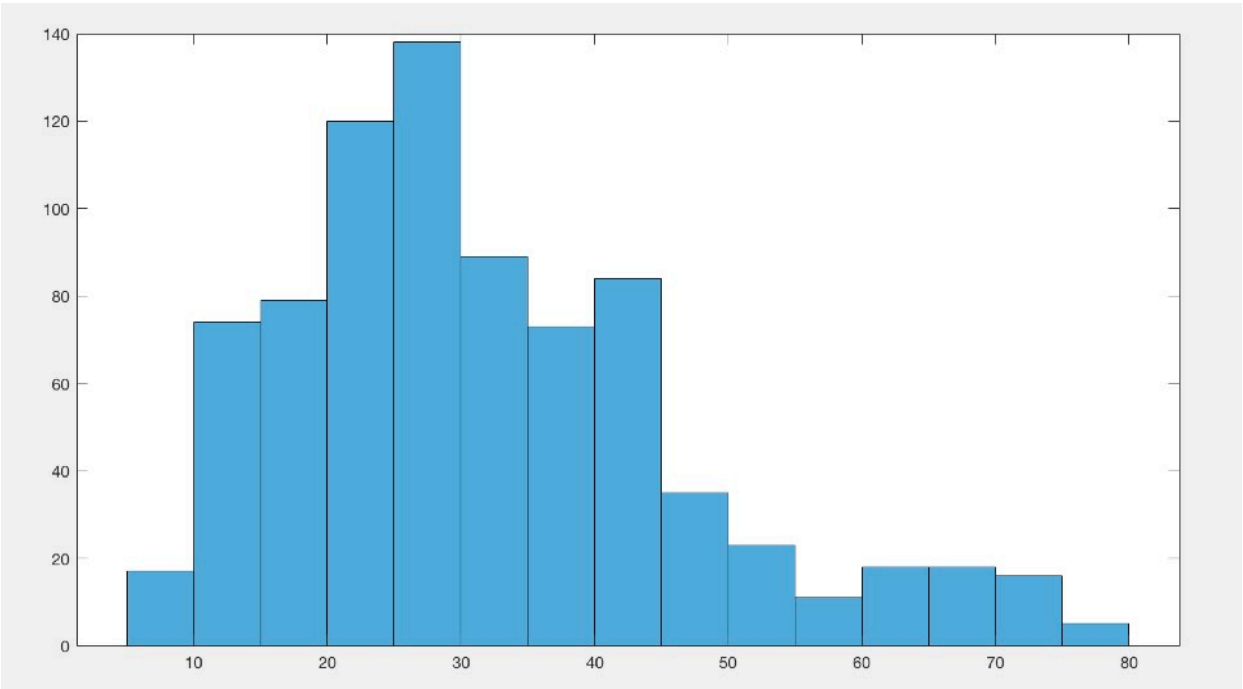




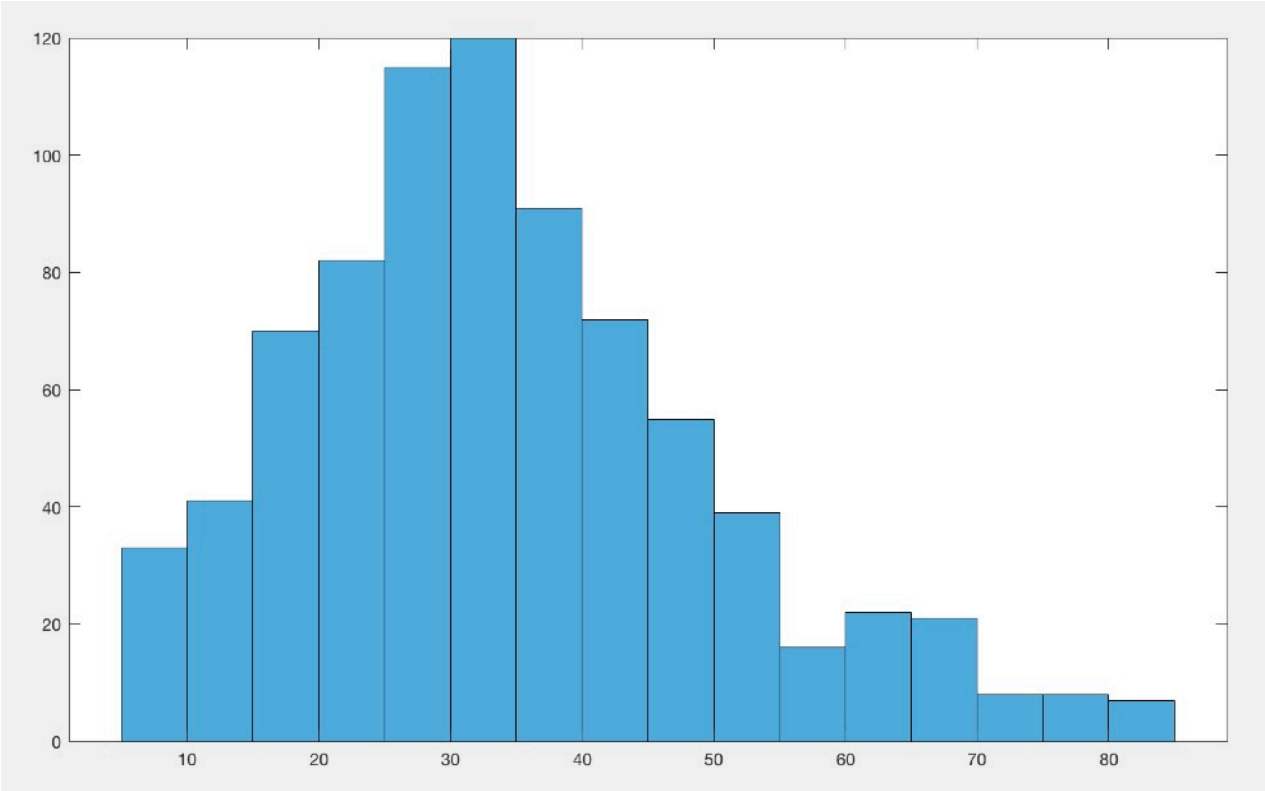
Tiempo pen-down:



Número máximos y mínimos de x:

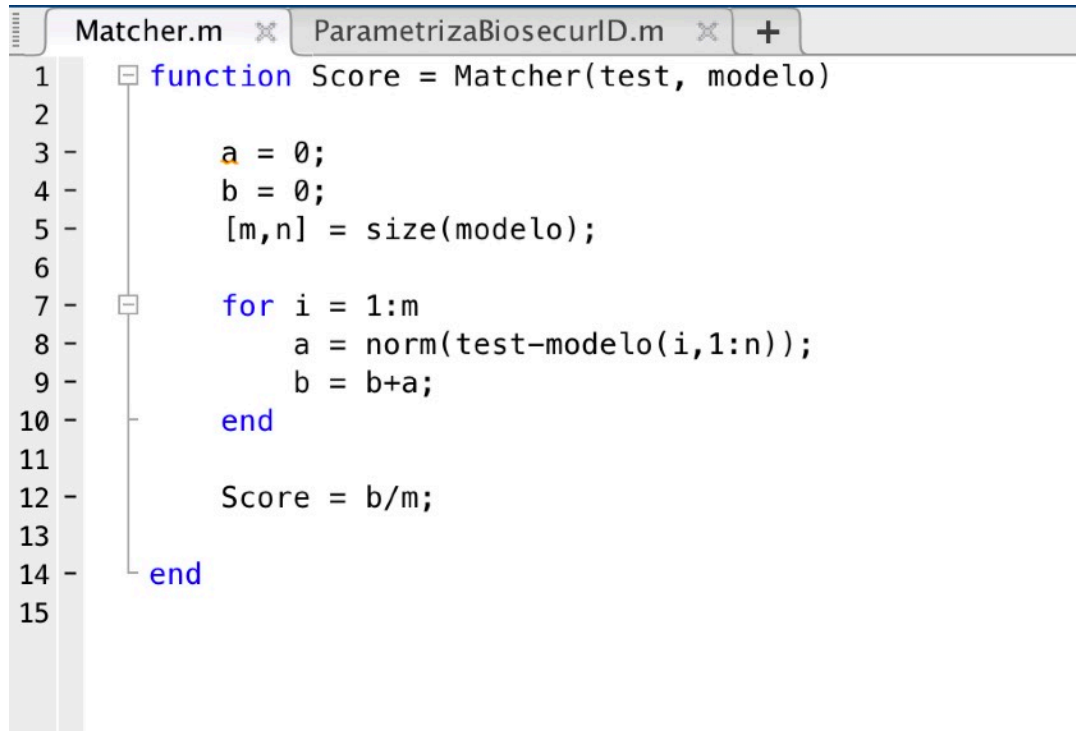


Número de máximos y mínimos de y:



### 3. Calcular el score entre las firmas registradas de un usuario y la firma de test con la distancia euclidea.

Lo primero sería crear una función matcher que calcule la distancia entre el vector que pasamos de las características de la firma de test, y una matriz con varias firmas, obteniendo una puntuación media de las distintas distancias obtenidas.

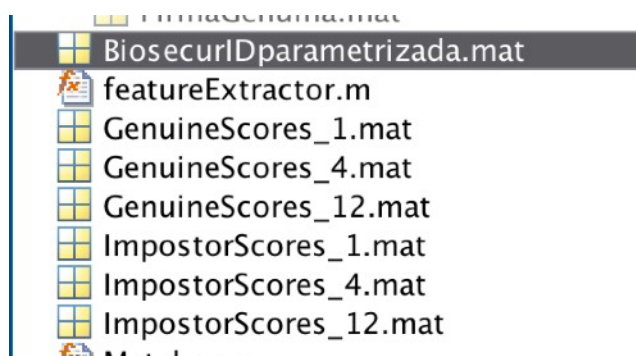


```
1 function Score = Matcher(test, modelo)
2
3     a = 0;
4     b = 0;
5     [m,n] = size(modelo);
6
7     for i = 1:m
8         a = norm(test-modelo(i,1:n));
9         b = b+a;
10    end
11
12    Score = b/m;
13
14 end
15
```

Luego se deben generar las matrices de Scores genuinos y Scores impostores, para ello vamos a modificar el archivo existente de “BiosecurIDparametrizada” introduciendo en él los bucles y código necesario para generar cada matriz y finalmente meter las sentencias de save que generan la matriz para cada caso.

```
save('BiosecurIDparametrizada','BiosecurIDparametrizada');
save('GenuineScores_1','GenuineScores_1');
save('GenuineScores_4','GenuineScores_4');
save('GenuineScores_12','GenuineScores_12');
save('ImpostorScores_1','ImpostorScores_1');
save('ImpostorScores_4','ImpostorScores_4');
save('ImpostorScores_12','ImpostorScores_12');
```

Se generarían las matrices anteriores:



Para formar las matrices de “GenuineScores” y de “ImpostorScores” como dijimos modificamos el archivo, y añadimos para cada matriz su código con los bucles y sentencias necesarias.

Para las GenuineScores recorreremos cada usuario, sacando una firma original suya y luego con bucles sacando el score de esa firma contra el resto de sus firmas.

```
ParametrizaBiosecurID.m  x  +
5  %Primera matriz de scores genuinos
6  GenuineScores_1=ones(50,15);
7  firma=1;
8  %Matriz 28x5 para obtener los vectores de características de cada una de
9  %las firmas de los usuarios
10 Modelo=ones(28,5);
11
12 for usuario=1:50
13     %Cargamos la firma 1 del usuario para usarla como firma test
14     if usuario<10
15         testID=load(['./DB/u100', num2str(usuario), 's000', num2str('1'), '_sg0000.mat']);
16     else
17         testID=load(['./DB/u10', num2str(usuario), 's000', num2str('1'), '_sg0000.mat']);
18     end
19     x_test=testID.x;
20     y_test=testID.y;
21     p_test=testID.p;
22     test_vect = featureExtractor(x_test,y_test,p_test);
23
24     for i=1:15
25         for sesion=1:4
26             for firma_orig=[1 2 6 7]
27                 if usuario<10
28                     BiosecurID=load(['./DB/u100', num2str(usuario), 's000', num2str(firma_orig), '_sg0000.mat']);
29                 else
30                     BiosecurID=load(['./DB/u10', num2str(usuario), 's000', num2str(firma_orig), '_sg0000.mat']);
31                 end
32
33                 x=BiosecurID.x;
34                 y=BiosecurID.y;
35                 p=BiosecurID.p;
```

```
ParametrizaBiosecurID.m x +
6
7 %Extracción de características:
8 - FeatVect = featureExtractor(x,y,p);
9
0 - Modelo(firma,1)=FeatVect(1);
1 - Modelo(firma,2)=FeatVect(2);
2 - Modelo(firma,3)=FeatVect(3);
3 - Modelo(firma,4)=FeatVect(4);
4 - Modelo(firma,5)=FeatVect(5);
5
6 - firma=firma+1;
7 - end
8 - end
9
0 %Calculamos el score de las firmas del usuario
1 - Score = Matcher(test_vect, Modelo);
2
3 %Introducimos el score en la matriz de genuine scores
4 - GenuineScores_1(usuario, i)=Score;
5
6 - i=i+1;
7 - firma=1;
8 - end
9 - end
0
1
```

Para las matrices de ImpostorScores tomamos una firma del usuario y la vamos a comparar contra las firmas del resto de usuarios, sacando los scores y formando estas matrices.

```

ParametrizaBiosecurID.m
26 %Primera matriz de scores impostor
27 ImpostorScores_1=ones(50,49);
28
29 for usuario=1:50
30 %Sacamos la firma 1 del usuario para usarla como modelo registrado
31 if usuario<10
32 BiosecurID_1=load(['./DB/u100', num2str(usuario),'s000', num2str(sesion)
33 else
34 BiosecurID_1=load(['./DB/u10', num2str(usuario),'s000', num2str(sesion)
35 end
36
37 x_1=BiosecurID_1.x;
38 y_1=BiosecurID_1.y;
39 p_1=BiosecurID_1.p;
40
41 %Extracción de características:
42 FeatVect_1 = featureExtractor(x_1,y_1,p_1);
43 contador=1;
44
45 %Sacamos la firma 1 del resto de usuarios para compararla contra la anterior
46 for usuario_2=1:50
47 if usuario_2~=usuario
48 if usuario_2<10
49 BiosecurID_2=load(['./DB/u100', num2str(usuario_2),'s000', num2str(sesion)
50 else
51 BiosecurID_2=load(['./DB/u10', num2str(usuario_2),'s000', num2str(sesion)
52 end
53
54 x_2=BiosecurID_2.x;
55 y_2=BiosecurID_2.y;
56 p_2=BiosecurID_2.p;
57

```

```

%Extracción de características:
FeatVect_2 = featureExtractor(x_2,y_2,p_2);

Modelo(contador,1)=FeatVect_2(1);
Modelo(contador,2)=FeatVect_2(2);
Modelo(contador,3)=FeatVect_2(3);
Modelo(contador,4)=FeatVect_2(4);
Modelo(contador,5)=FeatVect_2(5);

%Calculamos el score de la firma del usuario contra la primera del
%resto
Score = Matcher(FeatVect_1, Modelo);

%Introducimos el score en la matriz de impostor scores
ImpostorScores_1(usuario, contador)=Score;

contador=contador+1;
end
end
end

```

Con el uso de la función de “reshape” aplanamos las matrices de genuines scores e impostor scores, para que sean un vector y se le puedan pasar a la función de Eval\_Det.

```

>> Genuine_1 = reshape(GenuineScores_1,[1,750]);
>> Genuine_4 = reshape(GenuineScores_4,[1,600]);
>> Genuine_12 = reshape(GenuineScores_12,[1,200]);
>> Impostor_1 = reshape(ImpostorScores_1,[1,2450]);
>> Impostor_4 = reshape(ImpostorScores_4,[1,2450]);
>> Impostor_12 = reshape(ImpostorScores_12,[1,2450]);
>> |

```

Añadimos al path el directorio de las funciones.

Llamamos a la función “Eval\_Det” con los vectores de Genuine e Impostor para los casos de N=1, 4 y 12. Por cada llamada se obtienen tres valores.

```
% Impostor_1 = reshape(Impostor_1,1,1000);  
>> addpath('DetPlots')  
>> [EER1,DCF_opt1,ThresEER1]=Eval_Det(Genuine_1,Impostor_1,'b')
```

EER1 =

60

DCF\_opt1 =

0.0980

ThresEER1 =

55.2327

```
>> [EER4,DCF_opt4,ThresEER4]=Eval_Det(Genuine_4,Impostor_4,'b')
```

EER4 =

100

DCF\_opt4 =

0.1000

ThresEER4 =

1

```
>> [EER12,DCF_opt12,ThresEER12]=Eval_Det(Genuine_12,Impostor_12,'b')
```

EER12 =

100

DCF\_opt12 =

0.1000

ThresEER12 =

1

También al lanzar la función de “Eval\_Det” se abre la ventana para pintar la gráfica, pero en este caso parece que algo no es correcto por la gráfica que aparece.

