

Unidad 2

INTRODUCCIÓN A MATLAB

PARTE II

Contenido

- Funciones Gráficas
- Creación de Procedimientos y Funciones
- Depuración

FUNCIONES GRÁFICAS

PLOT

La principal que usaremos en el curso será plot. Su uso básico es:

`Plot(x,y)`

Representa puntos en el espacio, ubicados en cada valor de x en el eje de ordenadas con un valor de y en el eje de abcisas. Por defecto une los puntos.

Ejemplo

```
>> x=1:1:5;
```

```
>> y=2*x;
```

```
>>plot(x,y)
```

PLOT

Si queremos ver los puntos no unidos,
pondremos:

```
plot(x,y,'o')
```

Si queremos ver los puntos más la línea
pondremos:

```
plot(x,y,'o-')
```

Hold on

Puede ser que queramos ver dos gráficas superpuestas.

```
x=1:1:100;  
y1 =rand(1,100).*x;  
y2 =rand(1,100).*x;
```

Si ejecutamos

```
>>plot(x,y1);  
>>plot(x,y2); %Machacamos la gráfica anterior.
```

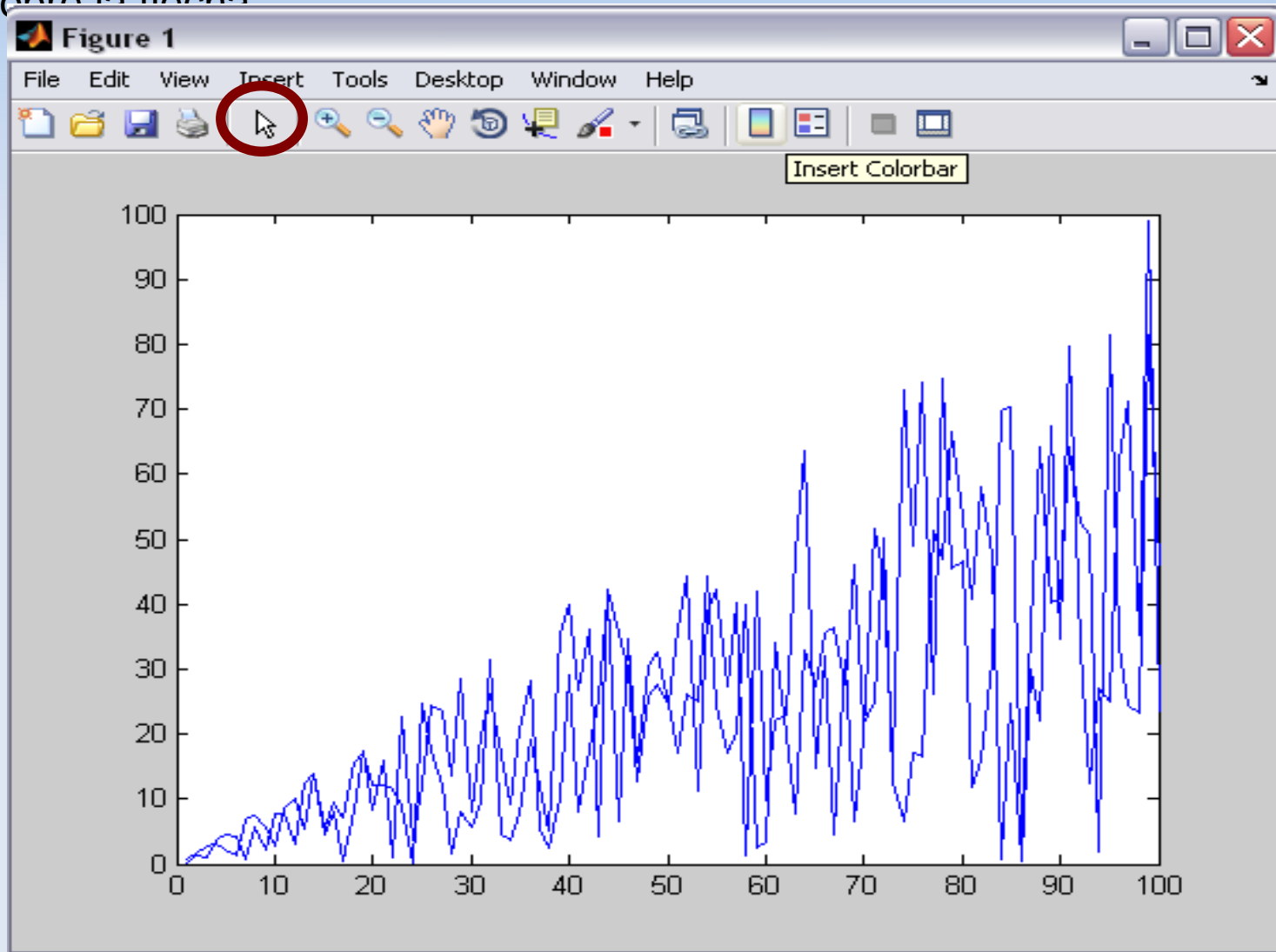
Para evitar este problema, podemos ejecutar la sentencia hold on (mantente) detrás del primer plot. A partir de ese momento, toda las gráficas se irán superponiendo.

```
>>plot(x,y1)  
>> hold on  
>> plot(x,y2)
```

Para desactivarlo basta con escribir hold off

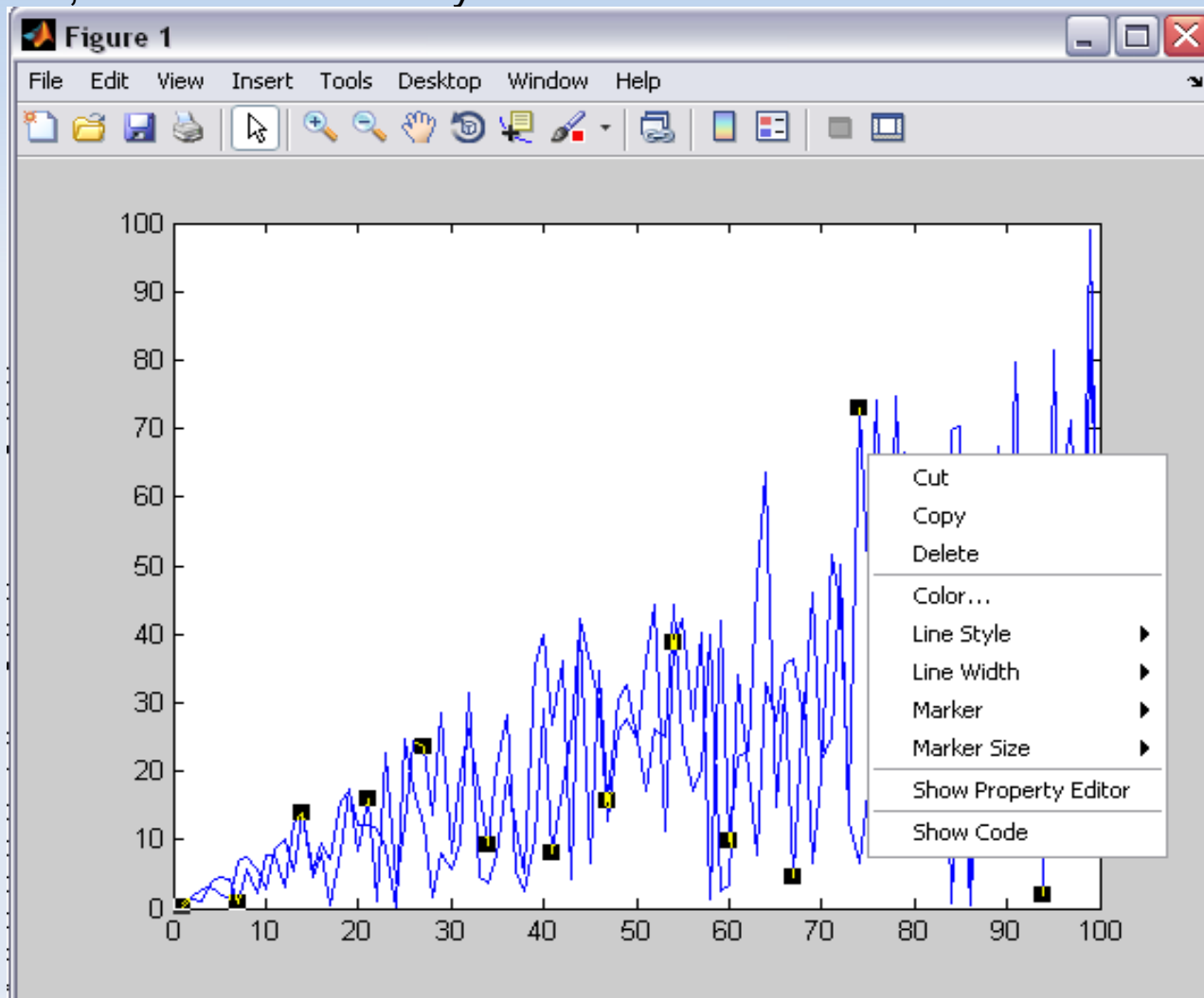
Maquetación de figuras

¿Cómo hemos puesto en color rojo la segunda gráfica? Cuando se muestran, pulsamos sobre la flecha



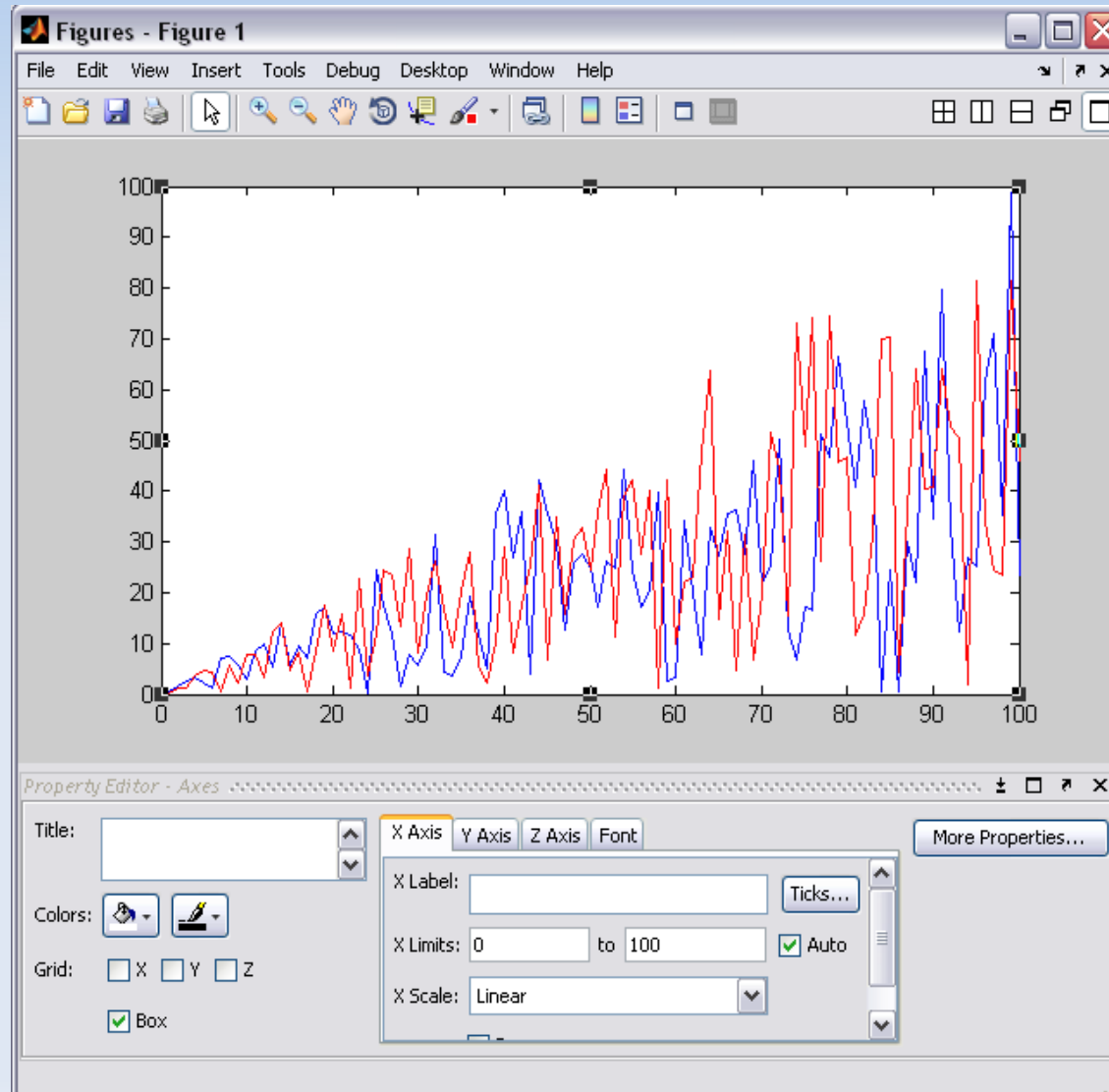
Maquetación de figuras

Con la flecha seleccionada pulsamos sobre la gráfica que queramos cambiar de color y pulsamos en el botón derecho y seleccionamos color. Vemos que también podemos cambiar el estilo, el ancho de línea y muchas más cosas.



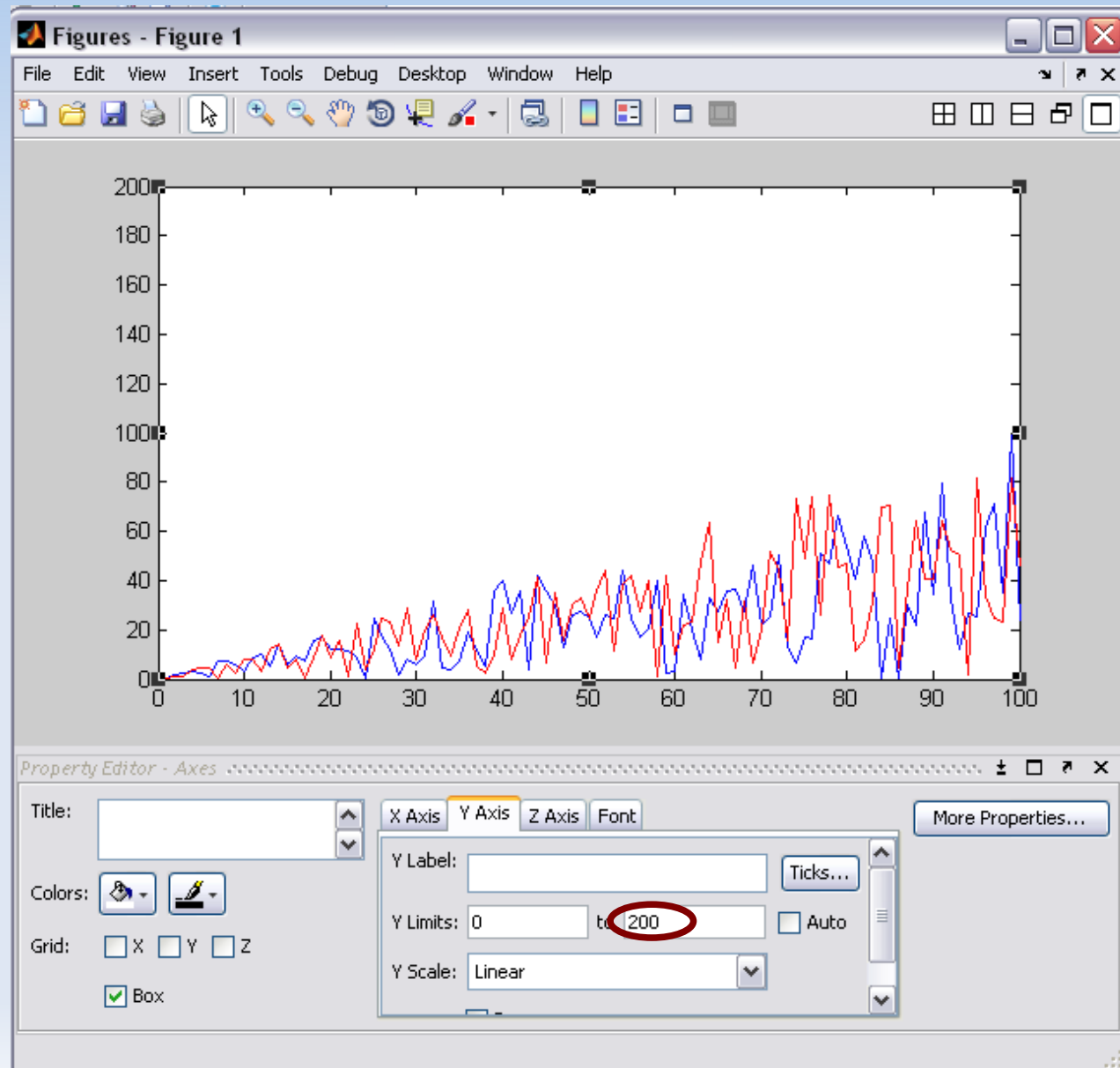
Maquetación de figuras

Vamos a cambiar ahora el rango de representación. Seleccionamos la gráfica en su conjunto, damos al botón derecho y pulsamos sobre show property editor



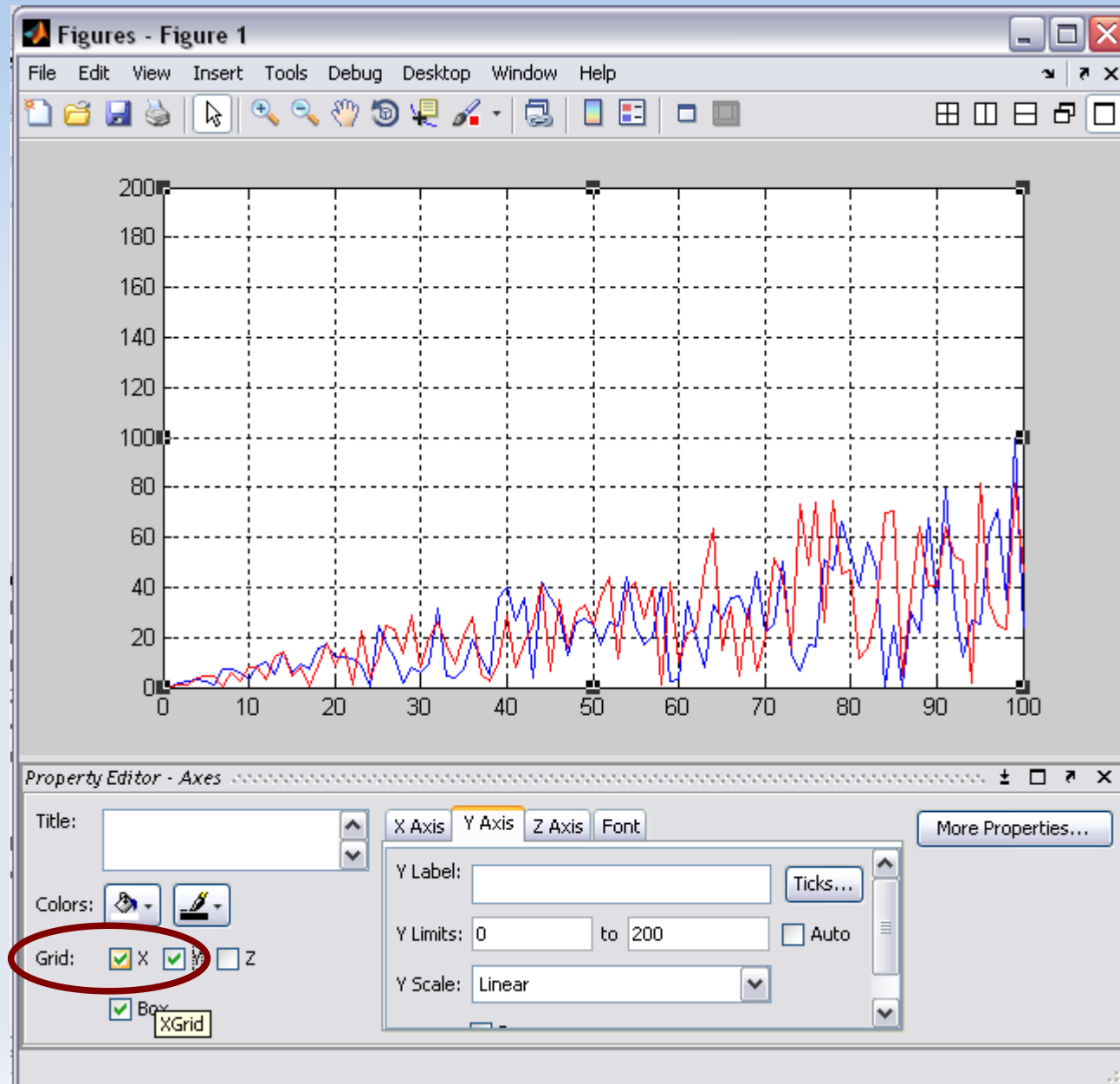
Maquetación de figuras

Pulsamos sobre Y Axis y cambiamos los límites (por ejemplo de 0 a 200)

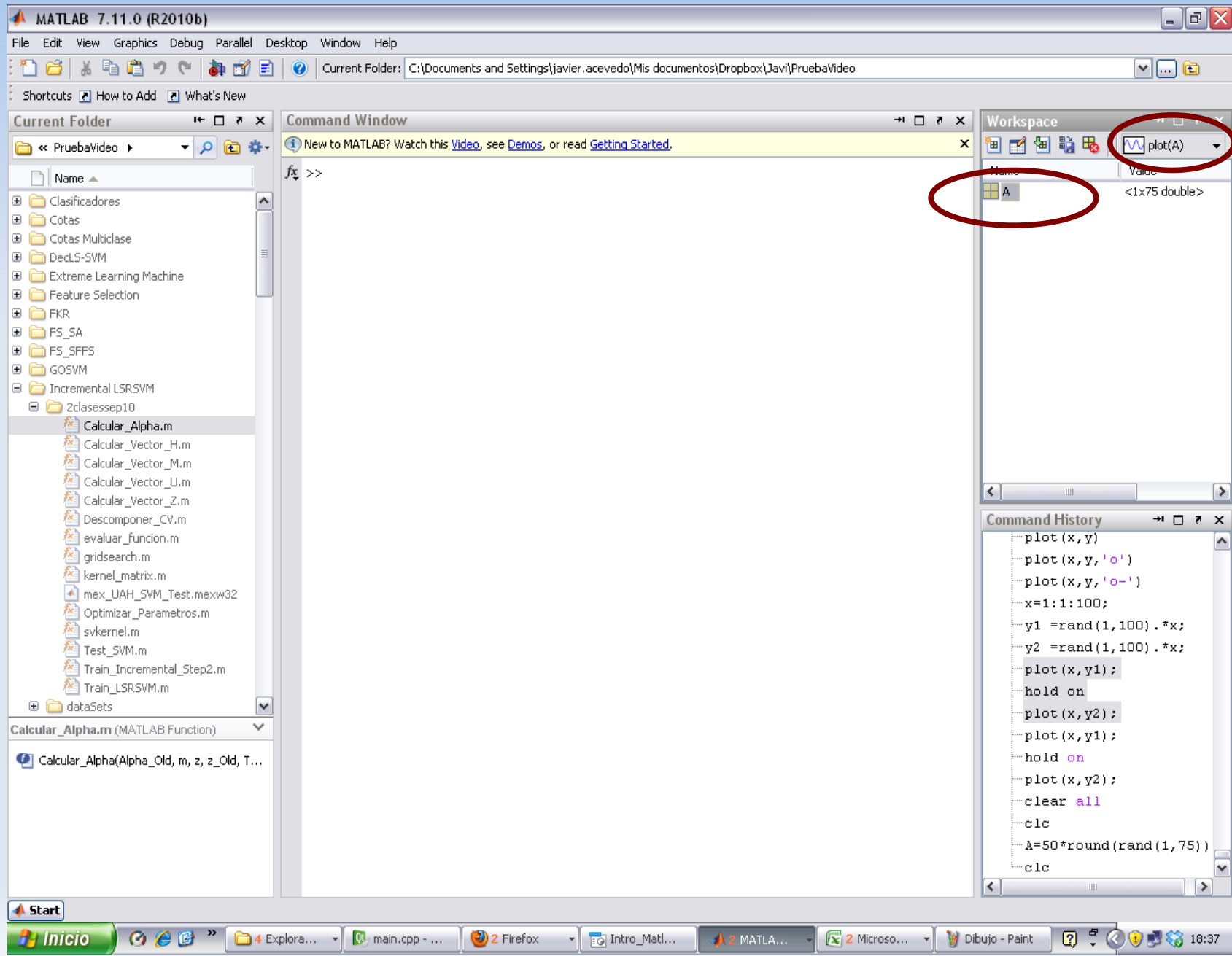


Maquetación de figuras

Por último vamos a añadir una rejilla.



Representación de un vector desde el workspace



Contenido

PROCEDIMIENTOS Y FUNCIONES

PROCEDIMIENTOS

Un procedimiento es una batería de instrucciones que se guarda en un fichero .m.

Al escribir el nombre del fichero ejecuta todas las instrucciones de igual forma que si estuviéramos en la ventana de comandos, por lo que dentro del procedimiento podemos hacer alusión a variables que se hayan declarado fuera.

Ejemplo:

Vamos a crear un problema y lo comprobaremos ejecutando varias veces un procedimiento

PROCEDIMIENTOS

En el Menú vamos a inciar el editor mediante la llamada a un nuevo procedimiento. Para ello, nos situamos en File-> New-> M-File

Podemos hacer un procedimiento que genere una secuencia de 100 números aleatorios entre el -5 y el 5 y vamos a encontrar la media de los números generados, la cantidad de ellos que son mayores o iguales que 0 y la cantidad que son menores que 0.

Cuando lo tengamos realizado guardamos el archivo como procedimiento1.mat.

Ahora desde la ventana de comandos ejecutamos procedimiento1 varias veces. Podemos apreciar que el efecto sobre la ventana de comandos es igual a copiar el código que hemos escrito y ejecutarlo desde la ventana de comandos.

IMPORTANTE: Solo podremos ejecutar el procedimiento si estamos en el mismo directorio de trabajo que el archivo .mat (o si hemos incorporado nuestro directorio a nuestro PATH).

PROCEDIMIENTOS

En una sala de baile hay 15 chicos y 15 chicas dispuestos en dos filas paralelas de manera que se formarán 15 parejas de baile. Sucede que la diferencia de altura entre el chico y la chica de cada pareja no supera los 10cm. Comprobar que si colocamos los mismos chicos y chicas en dos filas paralelas en orden creciente de alturas, también sucederá que la diferencia de alturas entre los miembros de las nuevas parejas así formadas no superarán los 10cm.

Abrir el editor

1. Crear un array aleatorio (Altura_Chicos) de quince elementos con mínimo 160 y máximo 200. Redondear los números.
2. Crear un array (Altura_Chicas) a partir del anterior que sume a cada elemento un número aleatorio comprendido entre más menos 10.
3. Comprobar que la diferencia de ambos arrays es menor que 10 en valor absoluto en todos los casos.
4. Ordenar los arrays anteriores por separado de menor a mayor
5. Comprobar que la diferencia de ambos arrays es menor que 10 en todos los casos

Guardamos el fichero como baile.m

Ahora ejecutaremos varias veces el programa para comprobar que siempre es cierto

FUNCIONES

Una función es un procedimiento opaco que acepta que pasemos variables como entradas y acepta variables como salidas.

Las variables declaradas dentro de la función (excepto las de salida) no son visibles desde el espacio de trabajo.

Su definición viene dada con la palabra clave `function` en la declaración del fichero `.m`.

Una función puede invocar a otras funciones que estén definidas en otros ficheros, siempre que se encuentren en el directorio de trabajo o definidas en el `PATH`.

Ejemplo: Editar el fichero `baile.m` y declararlo de esta forma

```
function Max_Dif=baile(N,maxima_altura)
```

Y modificar problema para que lo ejecute para cualquier número de parejas de baile y diferencia de altura máxima.

Desde la ventana de comandos. `d=baile(20,25);`

Contenido

DEPURACIÓN

DEPURACIÓN

Instrucciones



Set/Clear breakpoint: (F12) Coloca o borra un punto de ruptura en la línea en que está colocado el cursor



Clear all breakpoints:: Borra todos los puntos de ruptura



Step: (F10) Avanza un paso en el programa



Step in: (F11) Avanza un paso en el programa y si en ese paso se llama a una función, entra en dicha función



Step out: (MAY+F11) Si estamos dentro de una función ejecuta todas las instrucciones restantes que queden de la misma.



Continue: Continúa ejecutando hasta el siguiente punto de ruptura



Quit debugging: Termina la ejecución del debugger

DEPURACIÓN

Nuestro primer ejemplo de depuración.

Abrimos la función Ejemplo 1. Se trata de una función que representa dos sinusoides de distinta frecuencia, las representa y encuentra los puntos de cruce entre ambas señales.

Entradas:

A1 – Amplitud de la senoide 1. (El programa comprueba que es menor que 10)

W1- Pulsación de la senoide 1. (El programa comprueba que es menor que 10)

A2 – Amplitud de la senoide 2. (El programa comprueba que es menor que 10)

W2- Pulsación de la senoide 2. (El programa comprueba que es menor que 10)

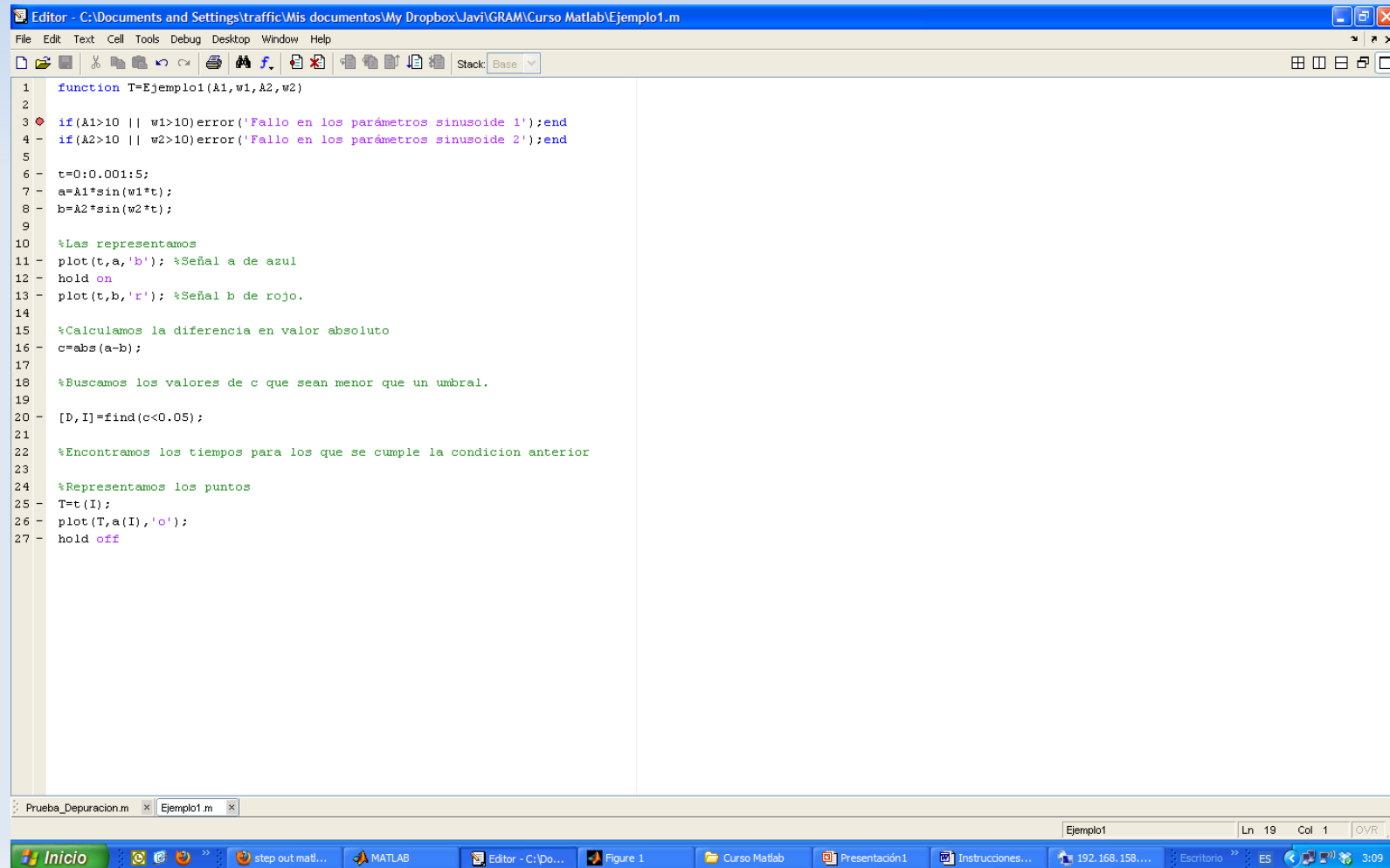
Salidas:

T- Vector que contiene los tiempos en los que se produce un cruce de señales.

DEPURACIÓN

Para poder entender el código, vamos a ejecutarlo paso a paso.

Lo primero que tenemos que hacer es añadir un punto de parada. Para eso nos situamos con el cursor en la primera instrucción y pulsamos (F12) o hacemos click sobre la izquierda al lado del número 3.



```
1 function T=Ejemplo1(A1,w1,A2,w2)
2
3 if(A1>10 || w1>10)error('Fallo en los parámetros senoide 1');end
4 if(A2>10 || w2>10)error('Fallo en los parámetros senoide 2');end
5
6 t=0:0.001:5;
7 a=A1*sin(w1*t);
8 b=A2*sin(w2*t);
9
10 %Las representamos
11 plot(t,a,'b'); %Señal a de azul
12 hold on
13 plot(t,b,'r'); %Señal b de rojo.
14
15 %Calculamos la diferencia en valor absoluto
16 c=abs(a-b);
17
18 %Buscamos los valores de c que sean menor que un umbral.
19
20 [D,I]=find(c<0.05);
21
22 %Encontramos los tiempos para los que se cumple la condicion anterior
23
24 %Representamos los puntos
25 T=t(I);
26 plot(T,a(I),'o');
27 hold off
```

DEPURACIÓN

Ahora desde la ventana de comandos vamos a ejecutar el programa:

```
>>T=Ejemplo1(8,7,5,3);
```

Vemos cómo el programa se queda detenido.

```
2
3  if(A1>10 || w1>10)error('Fallo en los parámetros senoide 1');end
4  if(A2>10 || w2>10)error('Fallo en los parámetros senoide 2');end
5
6  t=0:0.001:5;
```

Si ahora cambiamos al workspace vemos nuestras variables. Cambiamos al editor y pulsamos F10 (Ejecutamos el siguiente paso)

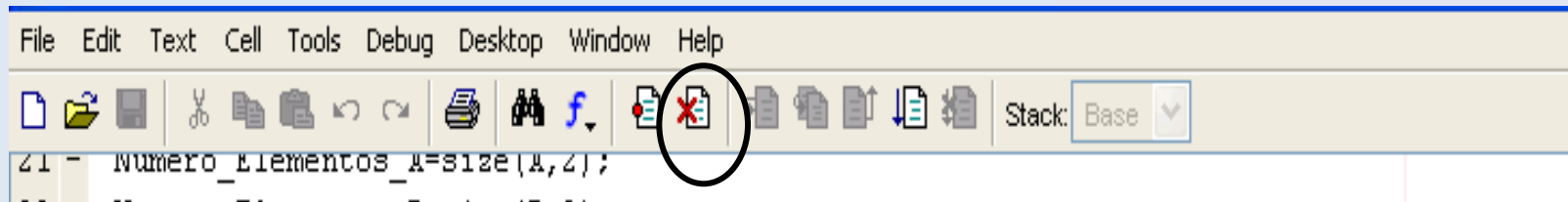
```
3  if(A1>10 || w1>10)error('Fallo en los parámetros senoide 1');end
4  if(A2>10 || w2>10)error('Fallo en los parámetros senoide 2');end
```

Seguimos avanzando hasta ejecutar la instrucción `c=abs(a-b);`

```
18  %Buscamos los valores de c que sean menor que un umbral.
19
20  [D,I]=find(c<0.05);
```

DEPURACIÓN

Una vez que hemos terminado de depurar un programa, nos aseguramos de quitar todos los posibles puntos de ruptura (breakpoints)

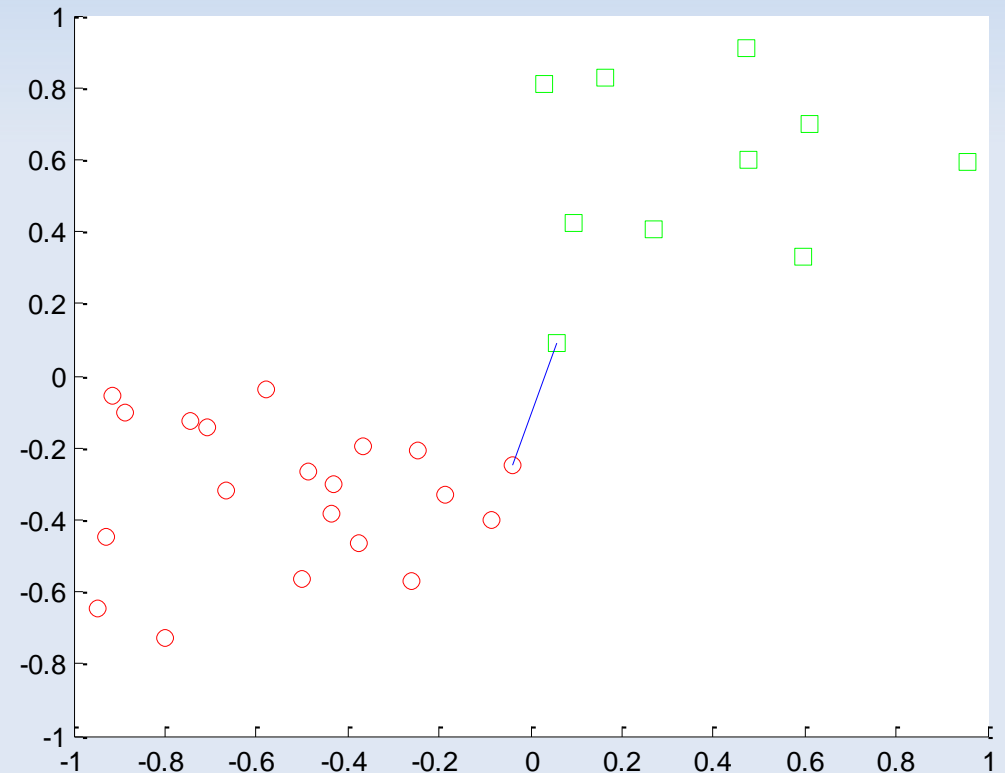


De esta forma, la próxima vez que invoquemos a la función no se detendrá.

DEPURACIÓN

Ejecución step-into

Ahora vamos a probar el ejemplo 2. En este caso se trata de pasar dos Conjuntos de puntos (A y B). El programa calcula la mínima distancia que existe entre los puntos del conjunto A con cada uno de los puntos del conjunto B y dibuja una línea entre dichos puntos.



Sin embargo, el programa que se os ha facilitado tiene fallos.
Vamos a depurarlo!!!

DEPURACIÓN

Antes de ejecutarlo vamos a definirnos nuestros conjuntos de puntos:

```
>>A=rand(2,20)-1; %Los valores de los puntos, tanto para X como para Y van de -1 a 0.
```

```
>>B=rand(1,10); %Los valores de los puntos, tanto para X como para Y van de 0 a 1.
```

Ponemos un punto de ruptura al principio de nuestro programa y ejecutamos:

```
>>resultado=Ejemplo2(A,B);
```

Vamos a ejecutar paso a paso hasta llegar a este punto

```
30 -         b=B(:,j);  
31         %Llamamos a la función calcular_distancia  
32 - ➡         distancia=calcular_distancia(a,b);  
33         D(i,j)=distancia;  
34 -     end  
--
```

DEPURACIÓN

Calcular_distancia es una función nuestra que es llamada desde Ejemplo2.

Ahora tenemos dos opciones.

- a) Ejecutar la función toda de golpe (F10).
- b) Meternos dentro de la función Calcular_distancia y ejecutar paso a paso los comandos de la misma

Para poder optar a esta segunda opción tenemos que “pasar dentro” (step into) para lo que pulsamos F11

```
30 -      b=B(:,j);  
31      %Llamamos a la función calcular_distancia  
32 - ➡      distancia=calcular_distancia(a,b);  
33 -      D(i,j)=distancia;  
34 -  
--
```

F11

DEPURACIÓN

Uso de múltiples breakpoints.



Una vez solventado el pequeño error que tenía la función Calcular_Distancia (hay que descubrirlo) seguimos ejecutando en nuestra función Ejemplo 2. Sin embargo, existe un problema:

```
for i=1:Numero_Elementos_A
    a=A(:,i);
    for j=1:Numero_Elementos_B
        b=B(:,j);
        %Llamamos a la función calcular_distancia
        distancia=calcular_distancia(a,b);
        D(i,j)=distancia;
    end
end
```

Si ejecutamos paso a paso tendremos que pasar por el bucle muchísimas veces. Para solventarlo, podemos fijar otro punto de ruptura en el programa y decir que ejecute hasta ese nuevo punto de ruptura.

DEPURACIÓN

Para que el programa continúe hasta el próximo breakpoint, pulsamos en el icono Continue o damos a F5.

```
14
15  if(size(A,1)~=2)error('La matriz A debe tener dimension 2');end
16
17 %Hacemos lo mismo con B
18
19 - if(size(B,1)~=2)error('La matriz A debe tener dimension 2');end
20
21 - Numero_Elementos_A=size(A,2);
22 - Numero_Elementos_B=size(B,2);
23
24 %Para cada punto de A calculamos su distancia con cada punto de B
25 %Vamos a almacenar el resultado en un array
26 %D(Numero_Elementos_AxNumero_Elementos_B)
27 - for i=1:Numero_Elementos_A
28 -     a=A(:,i);
29 -     for j=1:Numero_Elementos_B
30 -         b=B(:,j);
31 -         %Llamamos a la función calcular_distancia
32 -         distancia=calcular_distancia(a,b);
33 -         D(i,j)=distancia;
34 -     end
35 - end
36 %Ahora vamos a encontrar a qué par se corresponde la menor distancia
37 %encontrada.
38
39  [Minimos,Indices1]=min(D);
40 - [resultado,Indices2]=min(Minimos);
41
```

DEPURACIÓN

Otra opción que a veces resulta muy cómoda es colocarnos con el cursor hasta donde queremos que ejecute y dar a la opción

Debug->Go Until Cursor.

Una vez que tenemos estas herramientas, vamos a continuar ejecutando el programa.

Debug if error, Debug if warning.

Si ejecutamos el programa nos va a dar error. Sería muy interesante que justo antes de que nos diera el error y nos sacara del programa, éste se detuviera y nos avisara de que se va a producir un error. Vamos a quitar los breakpoints del programa y activar la opción

Debug->Stop if errors/warnings

Y activamos la casilla “Always stop if error”. Si ahora ejecutamos el programa vemos que se queda detenido donde se está produciendo el error. (debemos solucionarlo).

¿Funciona correctamente el programa? Encontrar el último fallo....