

Tema 1. Análisis de datos

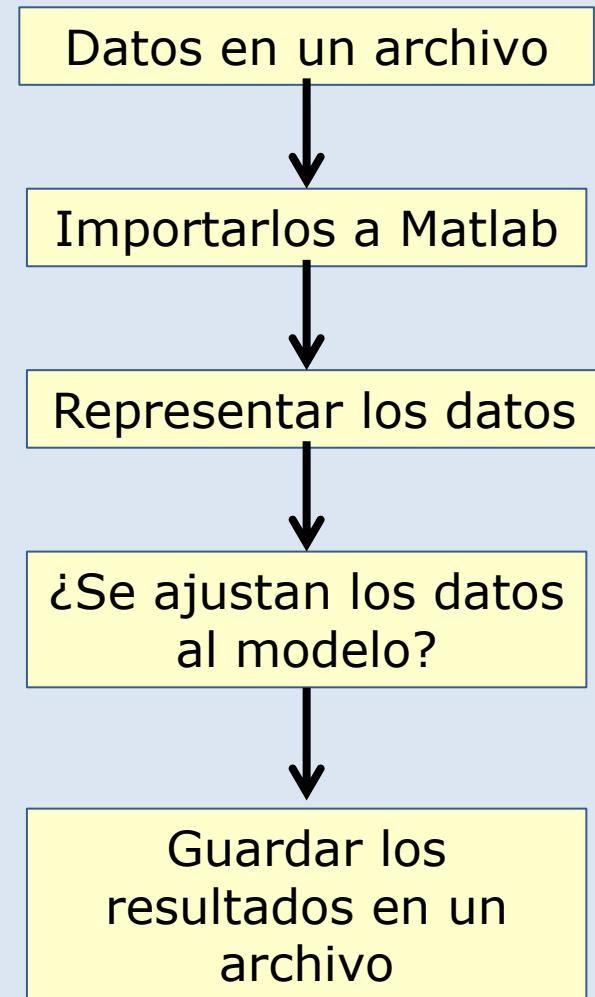
Métodos numéricos y
Simulación
Grado en Física

Contenido del tema

Problema: Recibimos un conjunto de datos en un archivo (creado por una persona o una máquina) y queremos saber si se ajustan a un modelo.

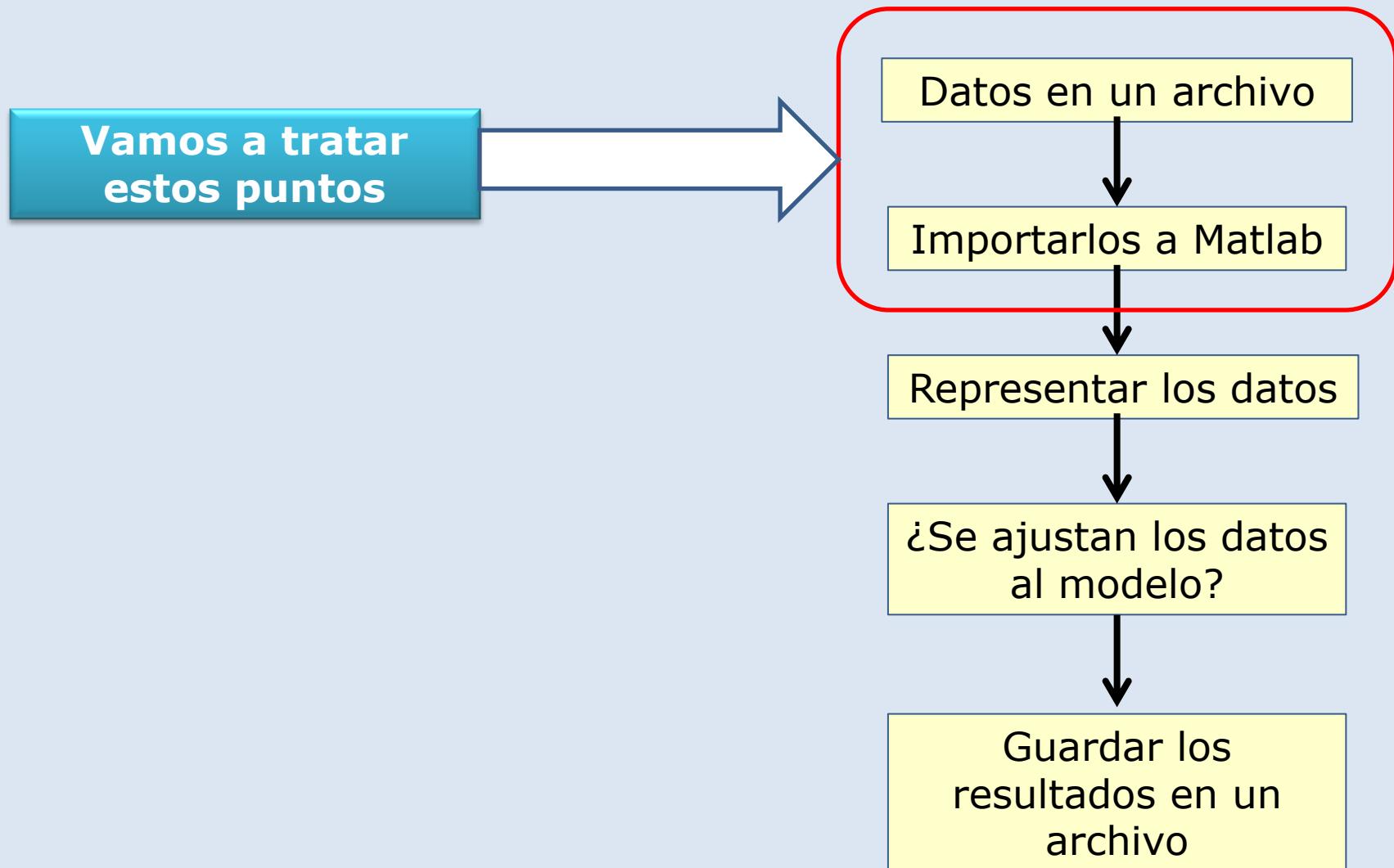
El esquema de trabajo a seguir es el que muestra la figura.

- Si sólo tenemos que hacer este trabajo una vez, podemos hacerlo manualmente.
- Si tenemos que hacer este trabajo repetitivamente, nos interesa programarlo.



Cómo usar Matlab para analizar datos.

Contenido del tema



Ejemplo: GuiaOndas.txt

Como ejemplo, vamos a importar los datos del archivo GuiaOndas.txt a Matlab.

Encontrarás este archivo en la WebCT de la asignatura.

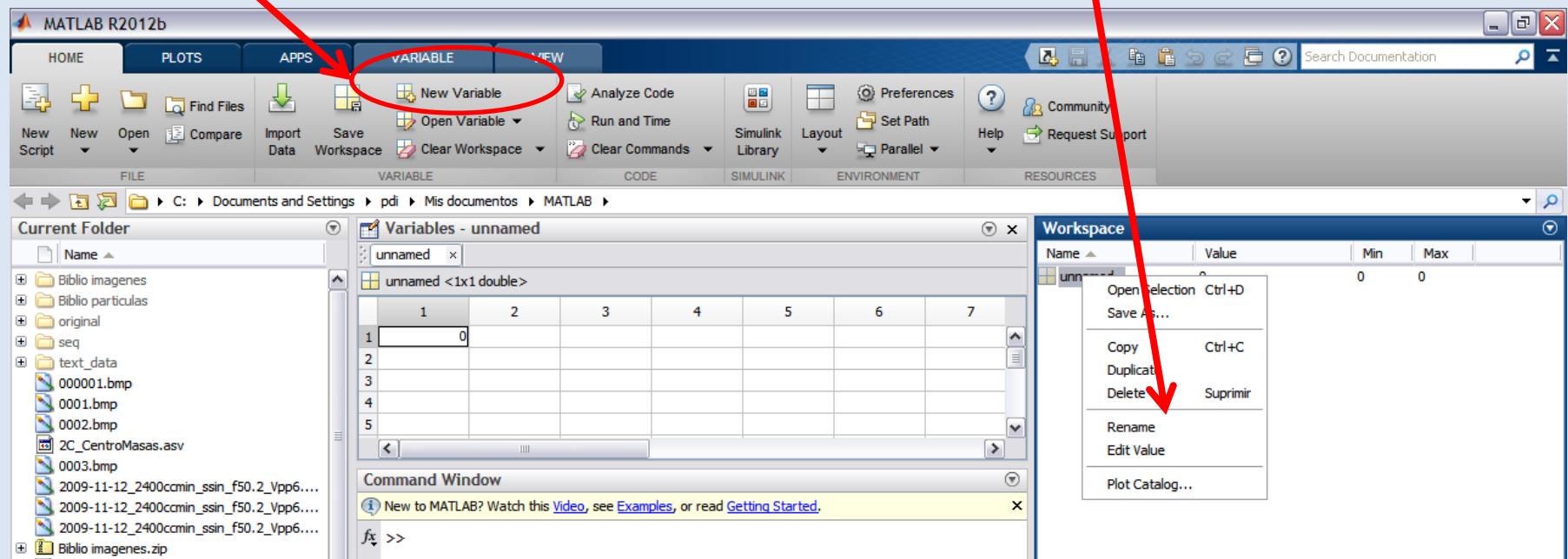
```
patos del campo eléctrico dentro de una guía de ondas
rectangular, tomados de una práctica de TEE
s es la posición de la sonda que mide el campo en mm
|E| es el módulo del campo eléctrico en unidades arbitrarias
s |E|
0 18.319
1 17.217
2 16.021
3 14.697
4 13.434
5 12.524
6 12.094
7 12.238
8 13.273
9 14.452
10 16.127
11 17.608
12 19.018
13 19.959
14 20.461
15 20.576
16 20.384
17 19.725
18 18.700
19 17.690
20 16.361
21 14.983
22 13.709
23 12.736
24 12.190
25 12.190
26 12.807
27 14.071
28 15.721
29 17.321
30 18.580
31 19.608
32 20.307
33 20.537
34 20.345
35 19.725
36 18.938
37 18.056
38 16.592
39 15.527
40 14.116
```

Importar al Array Editor: crear variables

La forma más simple de importar datos a Matlab es crear una variable y copiar a ella los datos que queremos usando el **Array Editor** y los comandos **Copiar** y **Pegar**

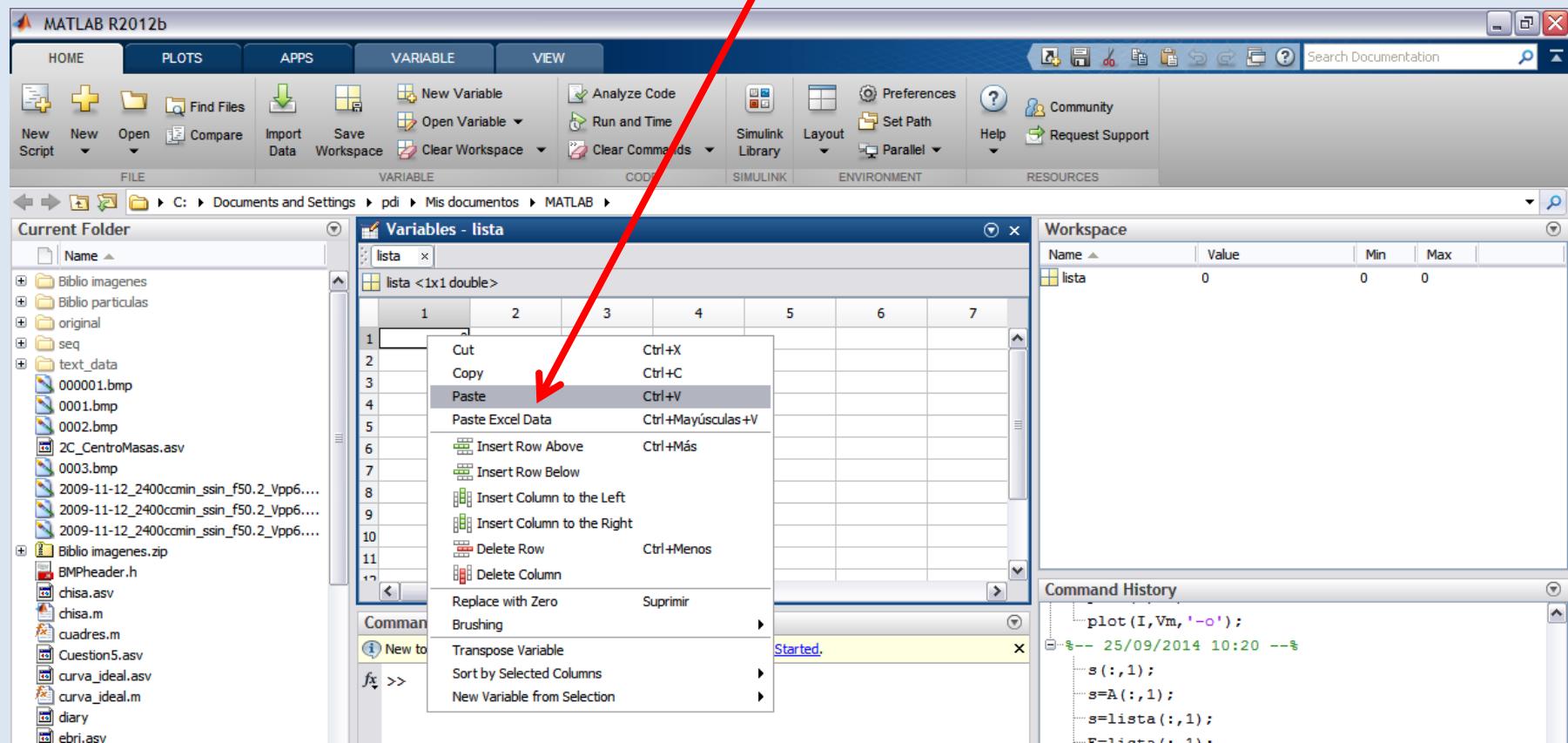
crear
variable

Pulsando el botón derecho del ratón sobre el ícono de la variable se nos ofrece la opción de cambiar su nombre



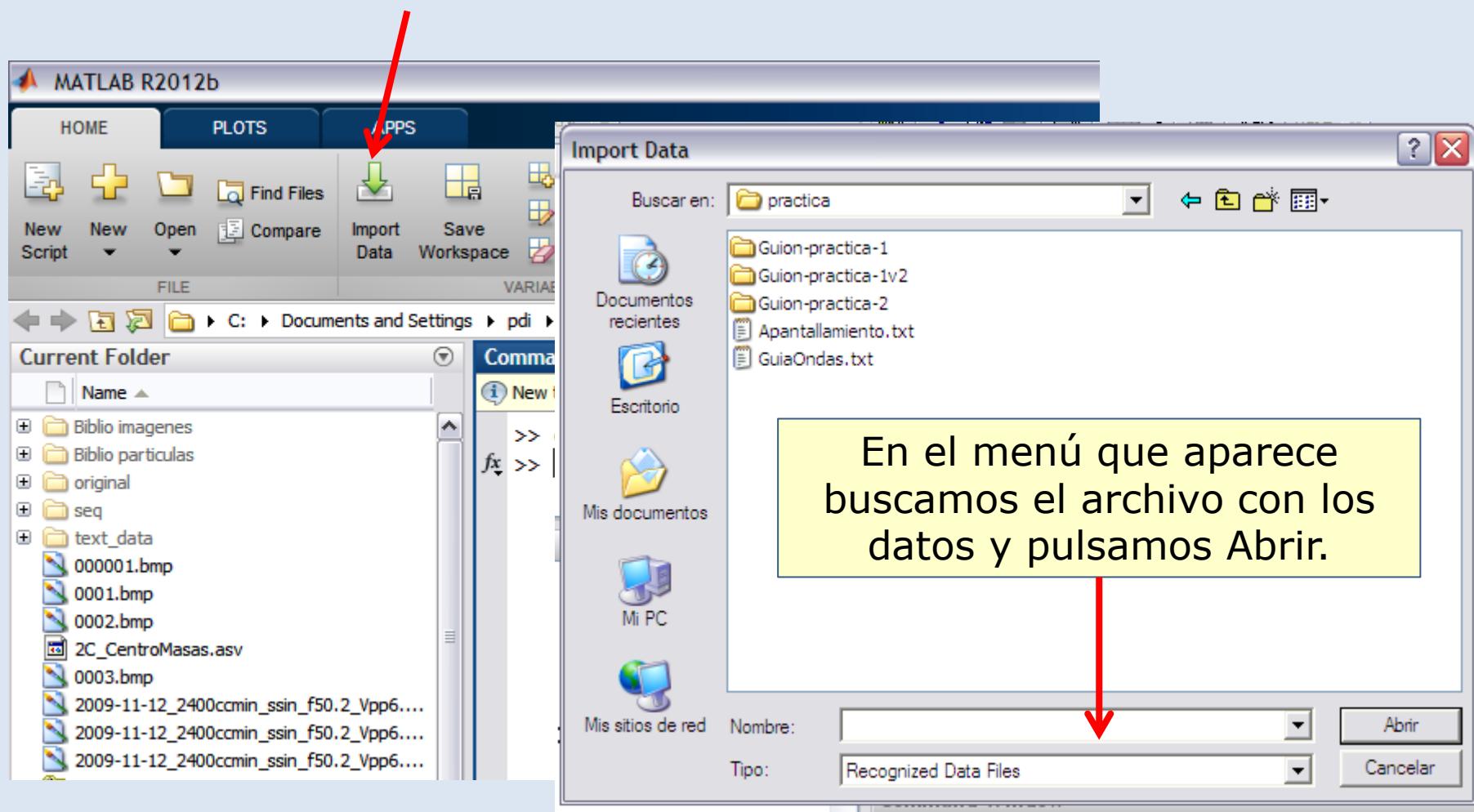
Importar al Array Editor: pegar valores

Abriendo la variable en el **Array editor** y pulsando con el botón derecho del ratón en una casilla se nos ofrece la opción de pegar valores en una variable.



Abrir el *Import Data Wizard*

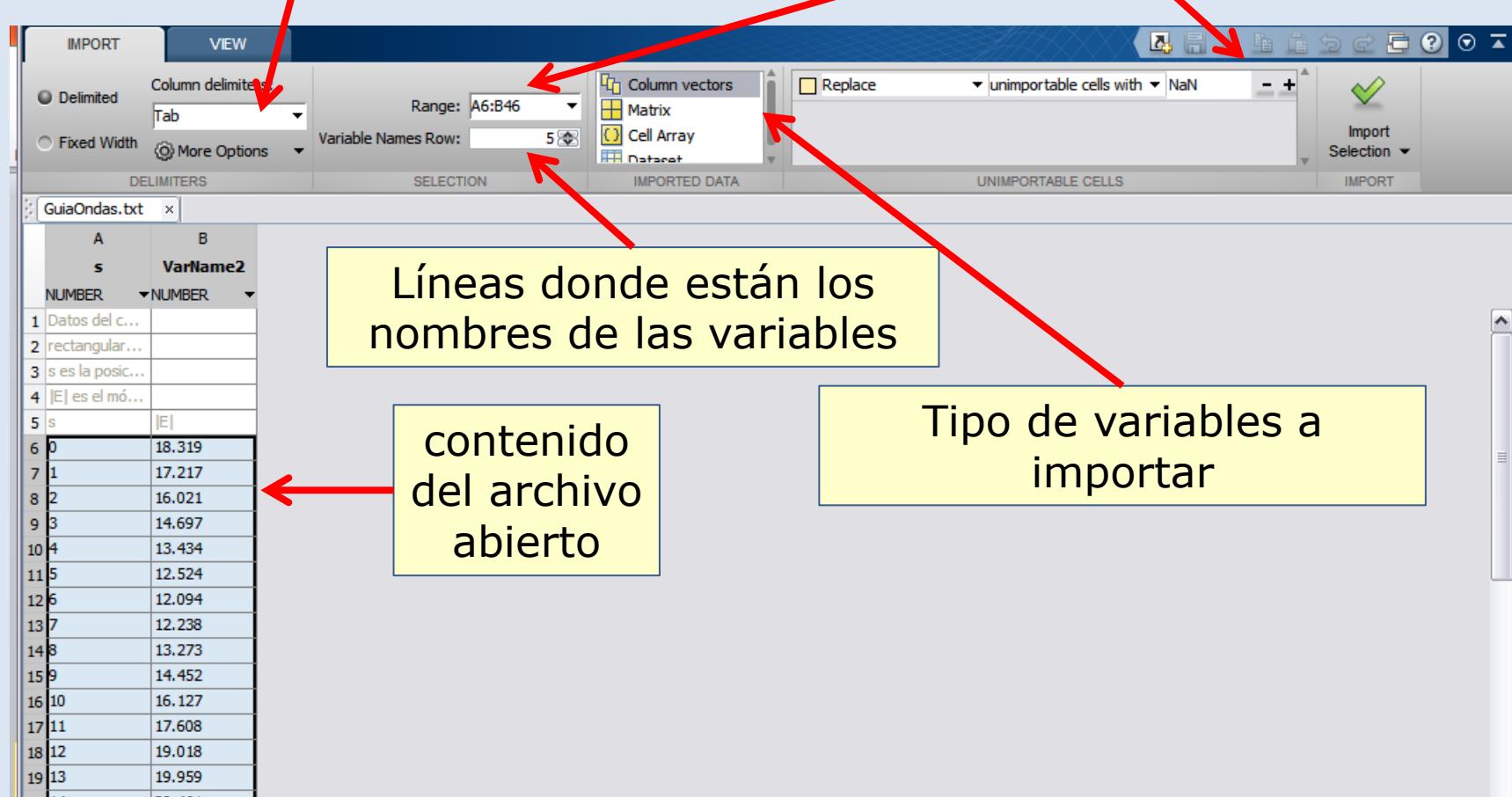
Cuando los datos están en un fichero en el disco duro, otra opción es ir a **Home/Import Data** y lanzar el **Import Data Wizard** de Matlab



Análisis del archivo de entrada

Tipo de separador entre columnas de datos

Rango donde están los valores de las variables



Elección del tipo de variables

El color de cada casilla indica si va a dar problemas de importación:



Puede importarse sin problemas.



Celda excluida.



Quizá de problemas.



No puede importarse.

Aquí le indicamos a Matlab qué queremos que haga con las casillas que dan problemas.

The screenshot shows the Matlab Import Data tool window. In the center, there's a preview table for a file named 'GuiaOndas.txt'. The first two columns are labeled 'A' and 'B'. The 'A' column has a dropdown menu open over the header 'NUMBER', showing options like 'TEXT (string)', 'NUMBER (double)', and 'DATE/TIME (serial date number)'. The 'B' column also has a dropdown menu open over its header 'NUMBER'. At the top of the window, there's a toolbar with various import options. On the right side of the toolbar, there's a section titled 'UNIMPORTABLE CELLS' with a dropdown menu set to 'Replace' and a dropdown next to it set to 'unimportable cells with NaN'. A red arrow points from the text in the bottom-left box to the 'NUMBER' dropdown in the preview table. Another red arrow points from the text in the top-right box to the 'Replace' dropdown in the toolbar.

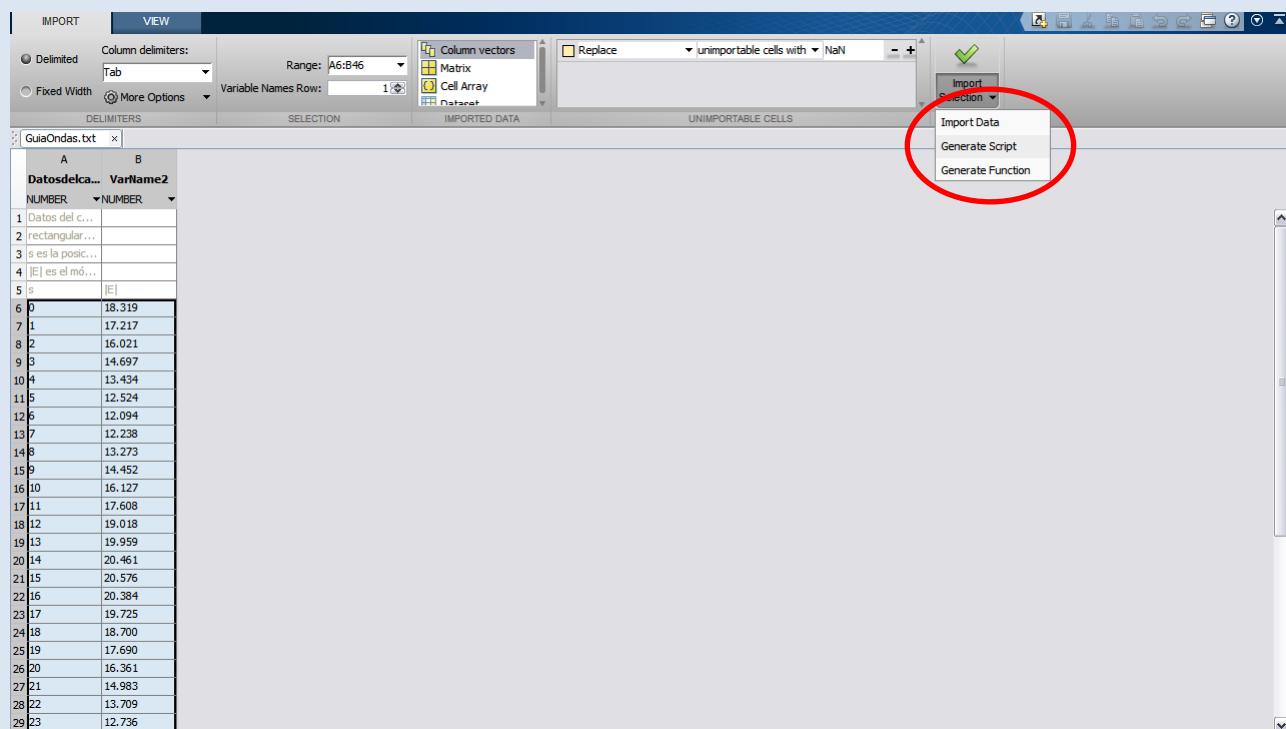
En la cabecera de cada columna podemos elegir el tipo de los datos: numérico (NUMBER), alfanumérico (TEXT) u otros. La elección de tipos que se nos ofrece depende del contenedor de datos que hayamos seleccionado.

Resultado de la importación

1) Variables en el Workspace.
Quizá tengamos que manipularlas para terminar de organizar los datos.

Name	Value	Min	Max
Datosdelcampoelctricoden	<41x1 double>	0	40
VarName2	<41x1 double>	12.0940	20.5760

2) Si aceptamos la opción **generate Script / generate Function**, Matlab crea también un fichero .m con las instrucciones necesarias para importar un fichero del mismo tipo en el futuro.



Programar la importación de datos.

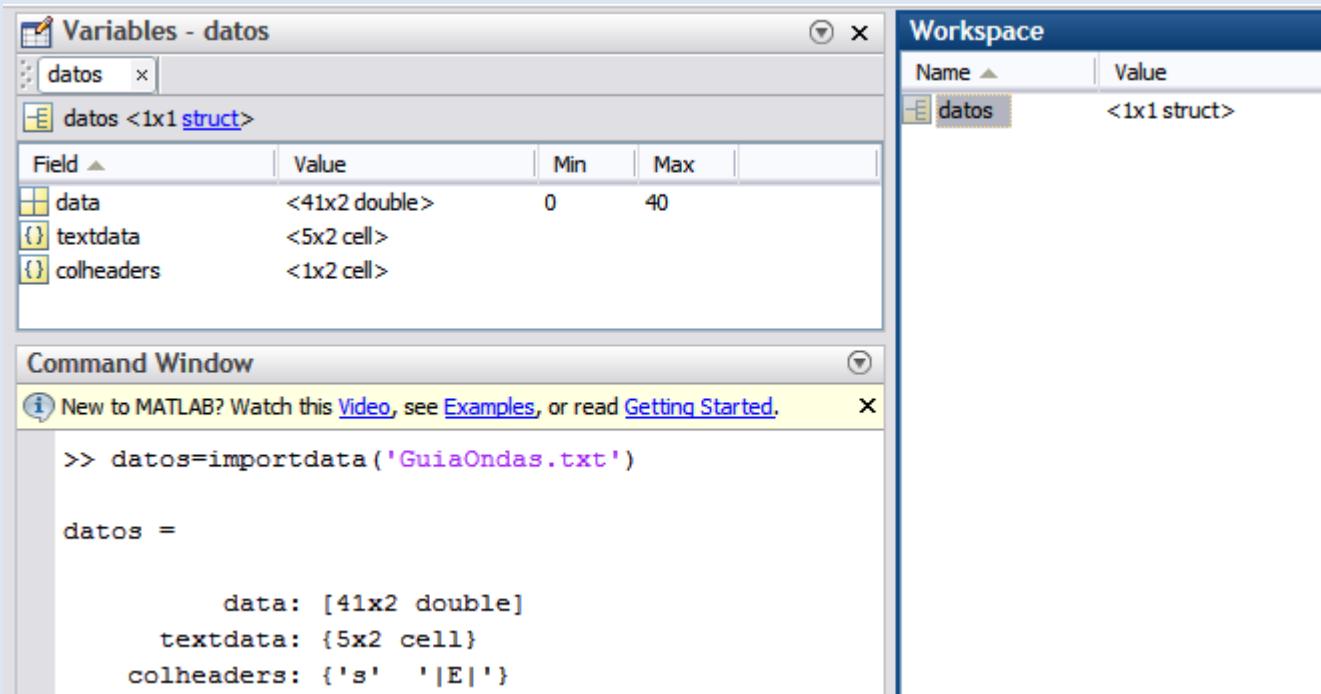
- A veces esta forma de programar la importación de datos no nos interesa porque:
 - El fichero de datos no tiene una estructura sencilla y no funciona bien con el *Import Data Wizard*.
 - La importación de datos se ha de hacer dentro de otro programa.
- Matlab tiene comandos para importar datos en un programa
 - Comandos de alto nivel. Hacen tareas complejas y toman decisiones por sí solos. Ejemplo: **importdata**
 - Funciones de bajo nivel. Hacen tareas sencillas. La tarea de pensar queda para nosotros.
 - Abrir y cerrar archivos: **open**, **close**
 - Leer de un archivo: **fgetl**, **textscan**, **fscanf**
 - Escribir a un archivo: **fprintf**

Función *importdata*

Funciona igual que el *Import Data Wizard*, pero se puede usar dentro de un programa.

datos=importdata(Nombrefichero)

Nombrefichero: nombre del fichero que queremos abrir
datos: una estructura con los mismos campos que el Importdata wizard (data, textdata,colheaders)



Ejemplo: Tabla-datos.txt (I)

Vamos a usar este nuevo archivo de datos como ejemplo porque no se importa fácilmente usando funciones de alto nivel.

Datos para la caída de tensión en una barra de cobre en función de la intensidad
La primera fila es la intensidad en amperios
Las columnas son la caída de potencial en micro-voltios
Resultados de TEII y FISII, curso 2009-2010.
Intensidad (A)
0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
d.d.p. (micro-v)
2.1 4.2 6.6 9 11.2 13.5 15.8 17.9 19.8 22.3
1 3 6 8 10 12 14 17 19 21
3 5 7.3 9 10.9 13.7 16.2 18.2 20.7 22.8
2.5 4.4 7 9 11.3 13.1 15.5 17.4 20 22.1
0.4 2.82 5.43 7.3 9.72 13.37 15.81 18.2 21.7 24.1
0.5 2.5 4.8 7.5 10 12.3 14.1 16.7 21.3 23.5
0.5 3 5.3 7.4 9.8 12.3 14.4 16.7 19 21.2
1.5 3.1 5.9 8.1 10.7 13.2 15 17.1 19.4 22.3
1.7 3.9 6.3 8.4 10.7 13.1 15.2 17.4 19.8 21.9
3.5 6 8.1 10.5 12.5 15.2 16.9 19.1 22.2 24.2
3 4.9 6.9 8.6 11.3 13.8 16.1 18.6 21.1 24.1
2 4.3 6.4 9.1 10.9 13.7 16 18 19.8 22.3
2.4 4.9 7.3 9.7 11.9 13.9 16.1 18.3 20.6 23
2.8 5 7.1 9.4 11.4 13.5 15.5 17.8 20.1 22.2

Import - D:\docencia\clases\materiales recurrentes\MNS\Tema_1_Analisis_Datos\practica\Ley_Ohm\Tabla-datos.txt

The screenshot shows the MATLAB Import tool window. The 'IMPORT' tab is selected. Under 'Column delimiters:', 'Delimited' is chosen with 'Tab' selected. The 'Range:' dropdown shows 'A8:J41'. The 'Variable Names Row:' dropdown shows '1'. On the right, there are options for 'Column vectors', 'Matrix', 'Cell Array', and 'Dataset'. Below these are 'Replace' and 'NaN' settings. A green checkmark icon with 'Import Selection' and 'IMPORT' buttons are also visible. The main workspace shows a table titled 'Tabladatos' with columns labeled A through J. The first few rows of data are displayed, corresponding to the text above.

	A	B	C	D	E	F	G	H	I	J
1	Tabladatos									
2	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
3	2.1	4.2	6.6	9	11.2	13.5	15.8	17.9	19.8	22.3
4	1	3	6	8	10	12	14	17	19	21
5	3	5	7.3	9	10.9	13.7	16.2	18.2	20.7	22.8
6	2.5	4.4	7	9	11.3	13.1	15.5	17.4	20	22.1
7	0.4	2.82	5.43	7.3	9.72	13.37	15.81	18.2	21.7	24.1
8	0.5	2.5	4.8	7.5	10	12.3	14.1	16.7	21.3	23.5
9	0.5	3	5.3	7.4	9.8	12.3	14.4	16.7	19	21.2
10	1.5	3.1	5.9	8.1	10.7	13.2	15	17.1	19.4	22.3
11	1.7	3.9	6.3	8.4	10.7	13.1	15.2	17.4	19.8	21.9
12	3.5	6	8.1	10.5	12.5	15.2	16.9	19.1	22.2	24.2
13	3	4.9	6.9	8.6	11.3	13.8	16.1	18.6	21.1	24.1
14	2	4.3	6.4	9.1	10.9	13.7	16	18	19.8	22.3
15	2.4	4.9	7.3	9.7	11.9	13.9	16.1	18.3	20.6	23
16	2.8	5	7.1	9.4	11.4	13.5	15.5	17.8	20.1	22.2

Uso de funciones de bajo nivel

- Necesitan que **abramos y cerremos** el fichero donde están los datos
- Requieren que les indiquemos el **formato** en el que nosotros queremos que estén los datos usando **sentencias de formato**.
- **Leen el fichero secuencialmente**, tal como nosotros lo haríamos “pasando el dedo por el texto”.
- Según el modo en que estén almacenados los datos en el archivo que queremos abrir, el archivo puede ser binario o de texto.
 - Los archivos de texto se reconocen porque se pueden “leer” su contenido con un editor de texto.
 - En este curso sólo trabajaremos con archivos de texto (los binarios están por encima del nivel de la asignatura).

Función *fopen*

Se usa para abrir un fichero.

id =*fopen(nombre,permisos)*

nombre: cadena de caracteres o variable alfanumérica con el nombre del fichero que queremos abrir.

id: variable numérica identifica el archivo en otros comandos del programa

permisos: el modo de acceso permitido al archivo para otros comandos del programa. Puede tomar, entre otros, los valores:

'rt': sólo lectura de texto. El archivo debe existir en el disco duro.

'wt': escritura de texto. Si el archivo no existe en el disco duro, crea uno nuevo; si ya existe, lo sobre-escribe.

'at': escritura. Si el archivo no existe en el disco duro, crea uno nuevo; si ya existe, escribe a partir del último de sus contenidos.

Función *fclose*

Se usa para cerrar un fichero.

`fclose(id)`

id: identificador del archivo que queremos cerrar.

Conviene cerrar los archivos que hemos abierto porque:

- Liberamos el espacio que ocupan en la memoria RAM
- Nos aseguramos de que los datos que están en los buffers de memoria del ordenador se escriben en el fichero.

Función *fgetl*

Lee una línea de un archivo, tomando como final de línea el retorno de carro (\n)

cadena = *fgetl*(id)

id: identificador del archivo del que quiero leer

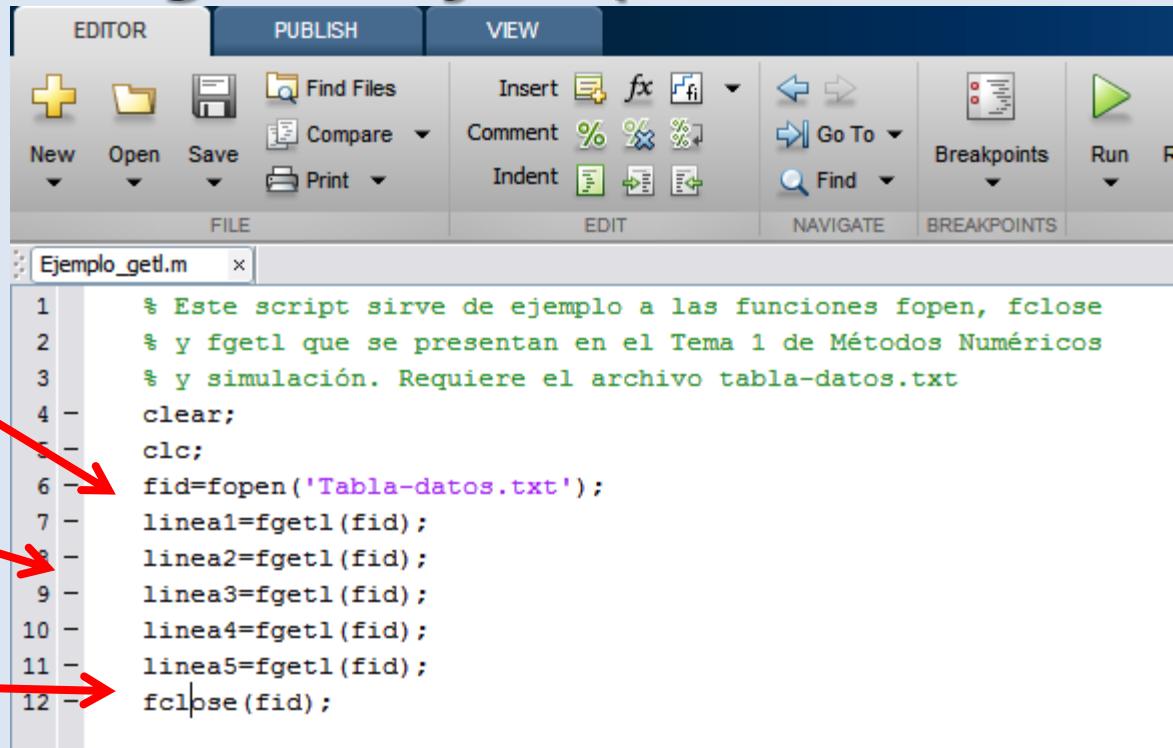
cadena: variable alfanumérica en la que se almacena la línea leída

fgetl considera todo el archivo como si fuera texto

Ventajas: No necesita sentencias de formato.

Inconvenientes: Si el contenido del archivo son datos numéricos, tenemos que extraerlos de la variable cadena

Función *fgetl*: ejemplo



The screenshot shows the MATLAB Editor window with the script file `Ejemplo_getl.m`. The code reads five lines from the file `'Tabla-datos.txt'` using the `fgetl` function. Red arrows point from callout boxes to specific lines of code: one arrow points to line 6 (`fid=fopen('Tabla-datos.txt');`), another to line 9 (`linea1=fgetl(fid);`), and a third to line 12 (`fclose(fid);`).

```
% Este script sirve de ejemplo a las funciones fopen, fclose  
% y fgetl que se presentan en el Tema 1 de Métodos Numéricos  
% y simulación. Requiere el archivo tabla-datos.txt  
  
clear;  
clc;  
fid=fopen('Tabla-datos.txt');  
linea1=fgetl(fid);  
linea2=fgetl(fid);  
linea3=fgetl(fid);  
linea4=fgetl(fid);  
linea5=fgetl(fid);  
fclose(fid);
```

abre el archivo
'tabla-datos.txt'

Leemos las 5
primeras líneas
del archivo

cierra el archivo

Así queda el
Workspace

Name	Value	Min	Max
ans	0	0	0
fid	3	3	3
linea1	'Datos para la caída de...'		
linea2	'La primera fila es la int...'		
linea3	'Las columnas son la ca...'		
linea4	'Resultados de TEII y ...'		
linea5	'Intensidad (A)'		

Función *textscan*

Lee datos de un archivo mientras se ajusten al formato que se le indica. Deja el puntero de lectura del archivo en el punto donde terminó de leer.

```
var=textscan(fid,'formato')
```

id: identificador del archivo del que quiero leer

var: es un **array de celdas** donde se guardan los datos leídos.

'formato': una cadena de caracteres con sentencias de formato.

Puede que tengamos que usar más comandos para organizar los datos contenidos en el array de celdas var.

Nota: el “puntero de lectura” es el dedo imaginario que vamos pasando por el archivo a leer.

Sentencias de formato

Indican a las funciones de entrada/salida de datos que tipo de datos esperamos leer o escribir. Tienen la forma:

%anchura.*decimales*tipo

tipo: un carácter que indica el tipo de notación a utilizar, entre otros:

c: carácter

s: cadena de caracteres

f: número fijo de decimales

d: numero entero

e: notación científica

Los tipos f y e tienen los campos opcionales:

anchura: el numero de dígitos a escribir, excluyendo el punto y el signo decimal.

decimales: el número de decimales.

El help de fprintf da más detalles sobre las secuencias de formato

Ejemplos de sentencias de formato

%6.5f	π	3.14159
%6.5f	$-\pi$	-3.14159
%6.5f	10π	31.41593
%6.5e	10π	3.14159e+001
%6.3f	10	10.000
%d	10	10
%s	Miguel Angel	Miguel Angel
%s	π	3.141593e+000

Función *textscan*: ejemplo

The screenshot shows the MATLAB IDE interface with the script file 'ejemplo_textscan.m' open. The code demonstrates how to use the *fopen*, *fclose*, *fgetl*, and *textscan* functions to read data from a file named 'tabla-datos.txt'. The code reads five lines of text and then uses *textscan* to read floating-point values from the fifth line. Red arrows point from three callout boxes on the left to specific lines of code: one arrow points to line 14 where *textscan* is used to read the first value from the fifth line; another arrow points to line 15 where *fgetl* is used to read the rest of the line; a third arrow points to line 16 where *textscan* is used to read all eight values from the fifth line.

```
% Este script sirve de ejemplo a las funciones fopen, fclose  
% fgetl y textscan que se presentan en el Tema 1 de Métodos Numéricos  
% y simulación. Tambien usa las sentencias de formato.  
% Requiere el archivo tabla-datos.txt  
clear;  
clc;  
% Importamos los datos del archivo  
fid=fopen('tabla-datos.txt');  
linea1=fgetl(fid);  
linea2=fgetl(fid);  
linea3=fgetl(fid);  
linea4=fgetl(fid);  
linea5=fgetl(fid);  
cellI = textscan(fid, '%f');  
linea=fgetl(fid);  
cellV = textscan(fid, '%f %f %f %f %f %f %f %f');  
fclose(fid);  
% Organizamos los datos  
% Se trata de pasar los datos de variables tipo celda  
% a variables numéricas  
I = cellI{1};  
V = zeros(34,10);  
for k=1:10  
    V(:,k)= cellV{k};  
end
```

Lee la fila de valores de la intensidad

Lee la siguiente línea de texto

Lee la tabla de valores del voltaje

Función *fscanf*

Lee datos de un archivo mientras se ajusten al formato que se le indica. Deja el puntero de lectura del archivo en el punto donde terminó de leer.

var = *fscanf(fid,'formato',[m n])*

id: identificador del archivo del que quiero leer

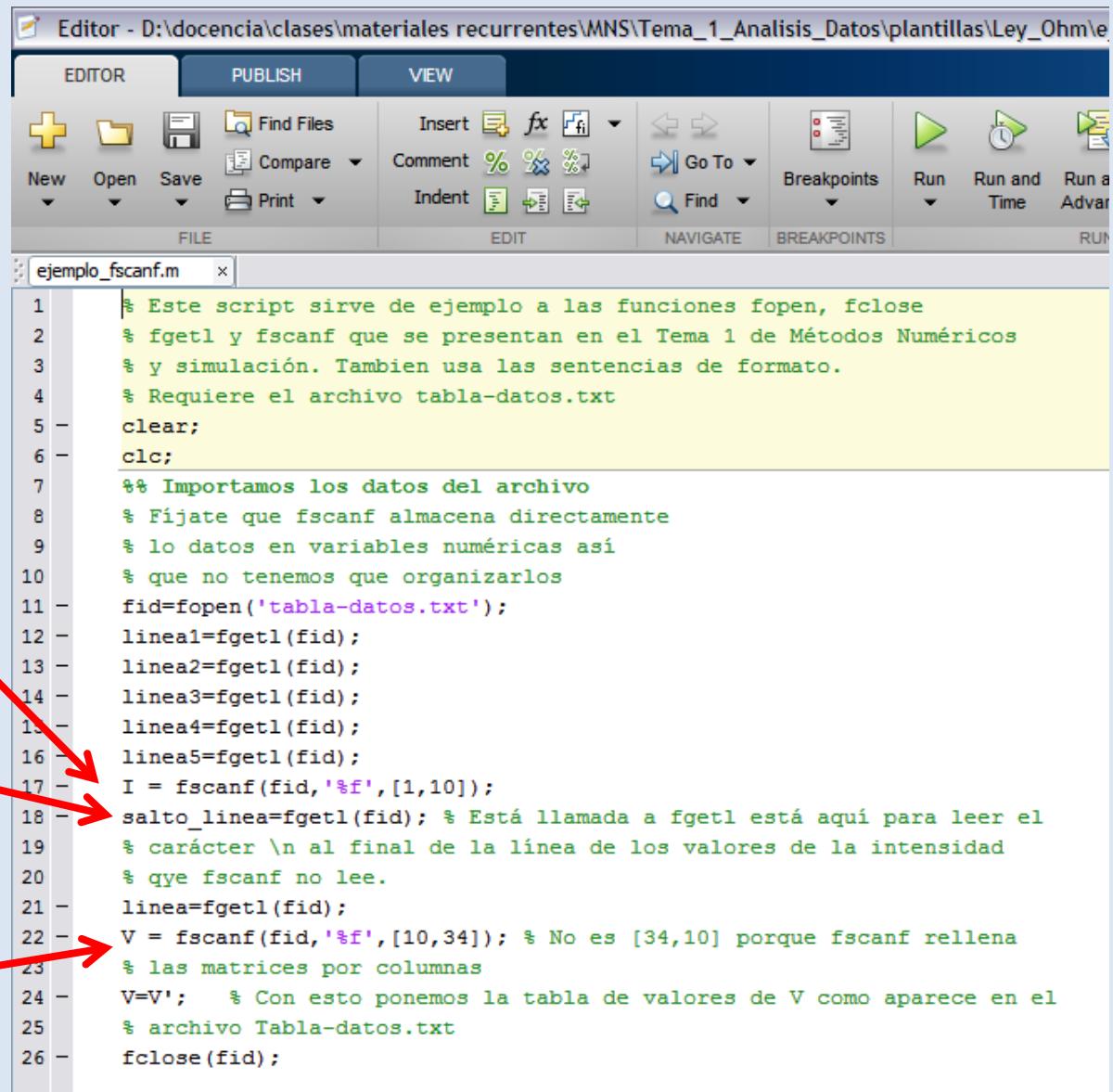
var: variable con **el mismo tipo de datos** que los que se indican en la sentencia de formato y en la que se almacenan los datos leídos

[m n] : $m \times n$ es la cantidad máxima de datos a leer.

fscanf trata de almacenar los datos en una matriz del tamaño que se le indica con **[m n]**.

Nota: el “puntero de lectura” es el dedo imaginario que vamos pasando por el archivo a leer.

Función *fscanf*: ejemplo



The screenshot shows a MATLAB editor window with the following details:

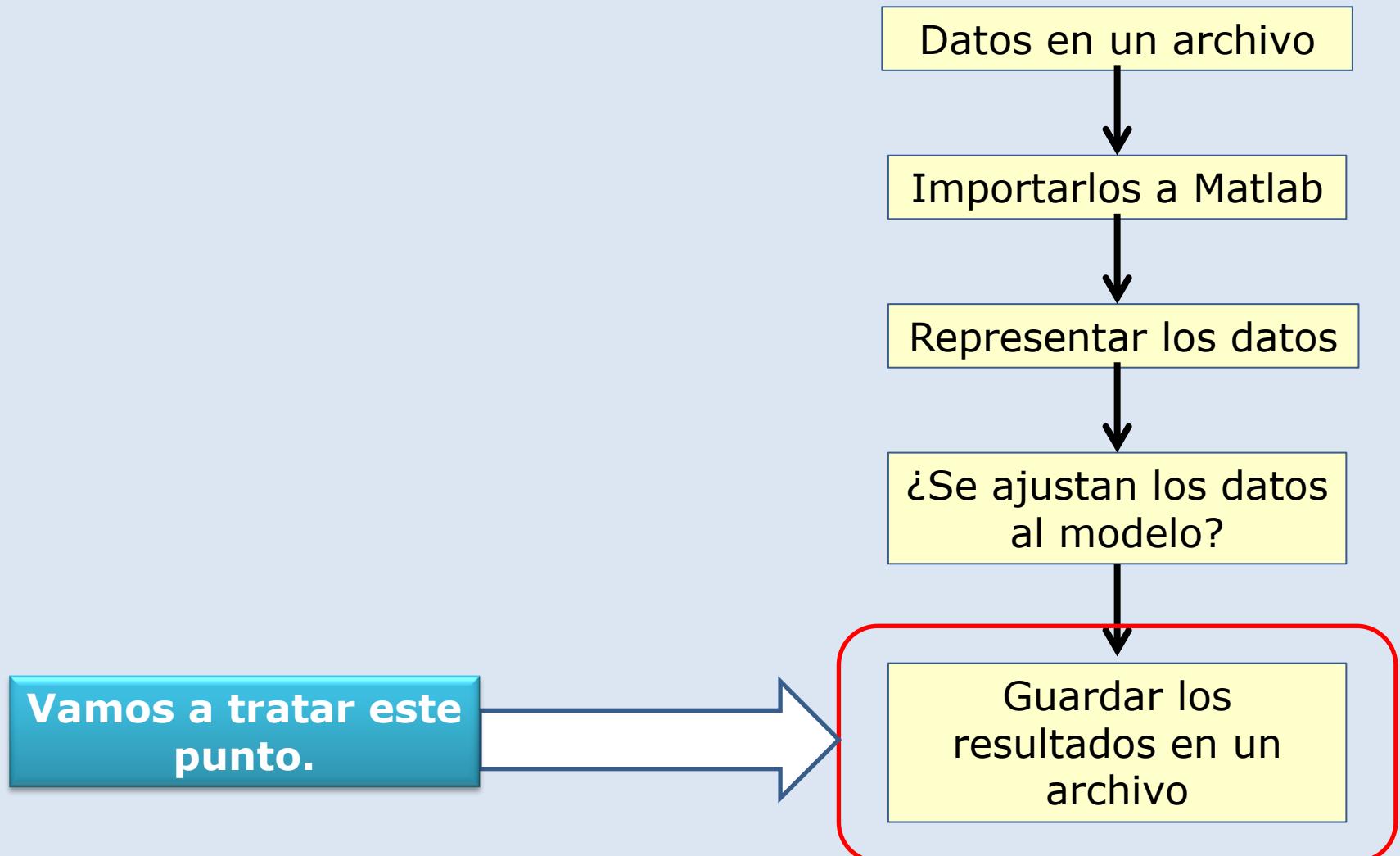
- Title Bar:** Editor - D:\docencia\clases\materiales recurrentes\MNS\Tema_1_Analisis_Datos\plantillas\Ley_Ohm\ejemplo_fscanf.m
- Toolbar:** Includes buttons for New, Open, Save, Find Files, Compare, Print, Insert, Comment, Indent, Go To, Breakpoints, Run, Run and Time, and Run Advanced.
- Code Area:** Displays the script content with line numbers on the left.

```
% Este script sirve de ejemplo a las funciones fopen, fclose  
% fgetl y fscanf que se presentan en el Tema 1 de Métodos Numéricos  
% y simulación. También usa las sentencias de formato.  
% Requiere el archivo tabla-datos.txt  
  
clear;  
clc;  
  
%% Importamos los datos del archivo  
% Fíjate que fscanf almacena directamente  
% los datos en variables numéricas así  
% que no tenemos que organizarlos  
  
fid=fopen('tabla-datos.txt');  
linea1=fgetl(fid);  
linea2=fgetl(fid);  
linea3=fgetl(fid);  
linea4=fgetl(fid);  
linea5=fgetl(fid);  
I = fscanf(fid, '%f', [1,10]);  
salto_linea=fgetl(fid); % Esta llamada a fgetl está aquí para leer el  
% carácter \n al final de la línea de los valores de la intensidad  
% que fscanf no lee.  
linea=fgetl(fid);  
V = fscanf(fid, '%f', [10,34]); % No es [34,10] porque fscanf rellena  
% las matrices por columnas  
V=V'; % Con esto ponemos la tabla de valores de V como aparece en el  
% archivo Tabla-datos.txt  
fclose(fid);
```

Annotations (Yellow Boxes):

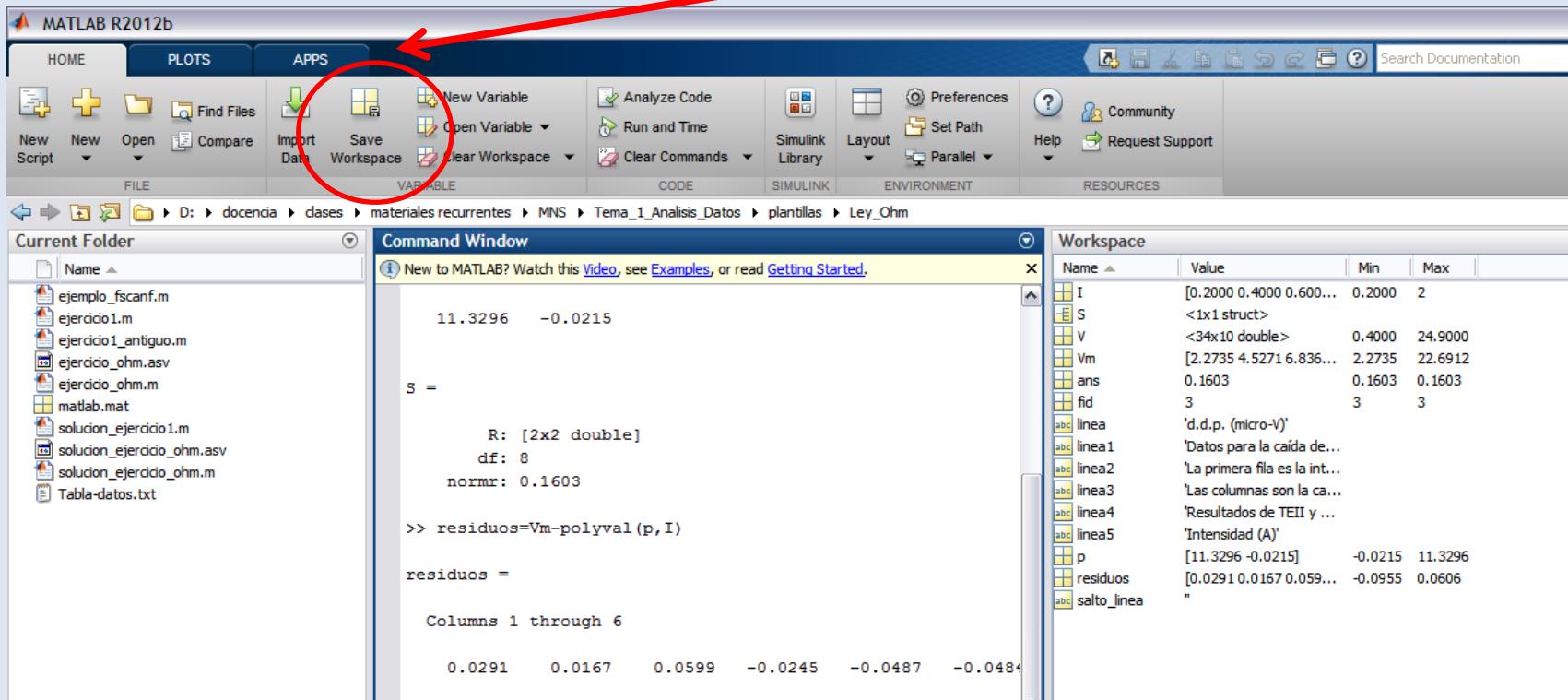
- Lee la fila de valores de la intensidad** (points to line 17)
- Lee la siguiente línea de texto** (points to line 18)
- Lee la tabla de valores del voltaje** (points to line 22)

Contenido del tema



Como guardar el Workspace

Si sólo queremos guardar las variables que tenemos en el Workspace, podemos hacer pulsando en **Save Workspace**



Las variables se guardan en un fichero con extensión .mat, que luego podemos abrir normalmente.

Escritura con formato. Función *fprintf*

Escribe datos con formato por pantalla o en un archivo

fprintf(fid,cadena de caracteres,variable1,variable2,...)

fid: identificador del archivo donde queremos escribir. Si no se incluye, la salida se hace por pantalla.

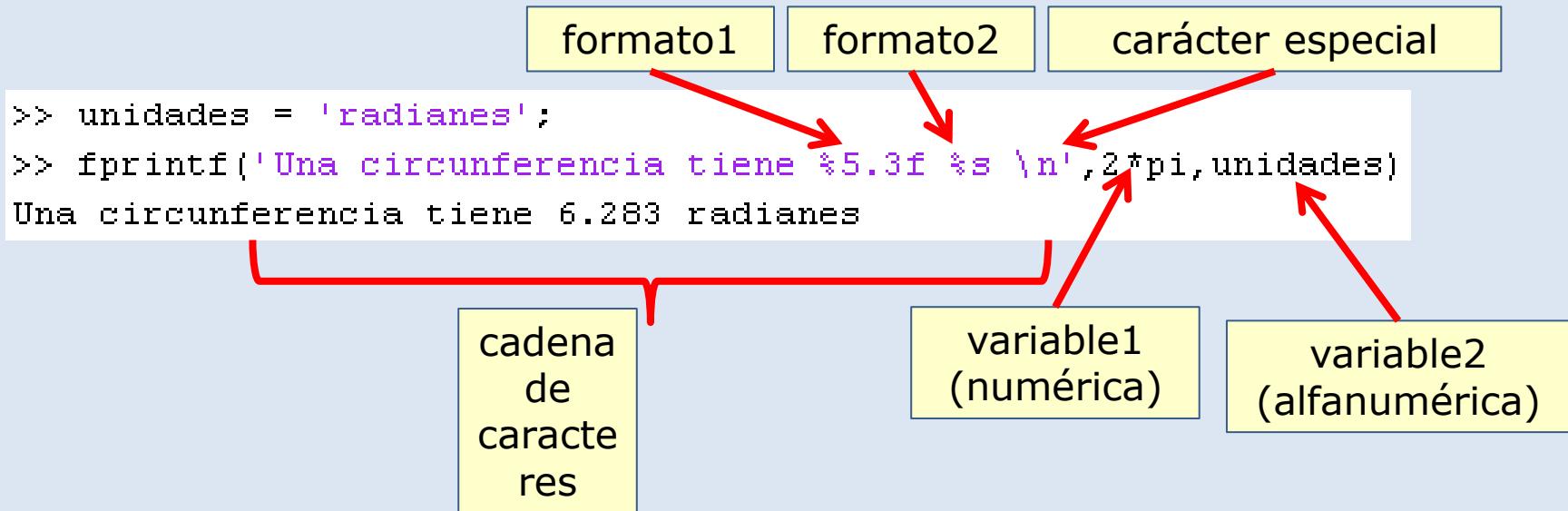
variable1, variable2,... variables cuyo valor queremos que aparezca en pantalla tras los caracteres de **texto1, texto2,...**

cadena de caracteres: Ha de tener la forma: '**texto1
formato1 texto2 formato2...**'

formato1, formato2,... sentencias de formato para las variables variable1, variable2,...

En la cadena de caracteres también podemos poner caracteres especiales

Ejemplo de uso de fprintf



También puede escribirse en una cadena en lugar de en pantalla usando el comando **sprintf**

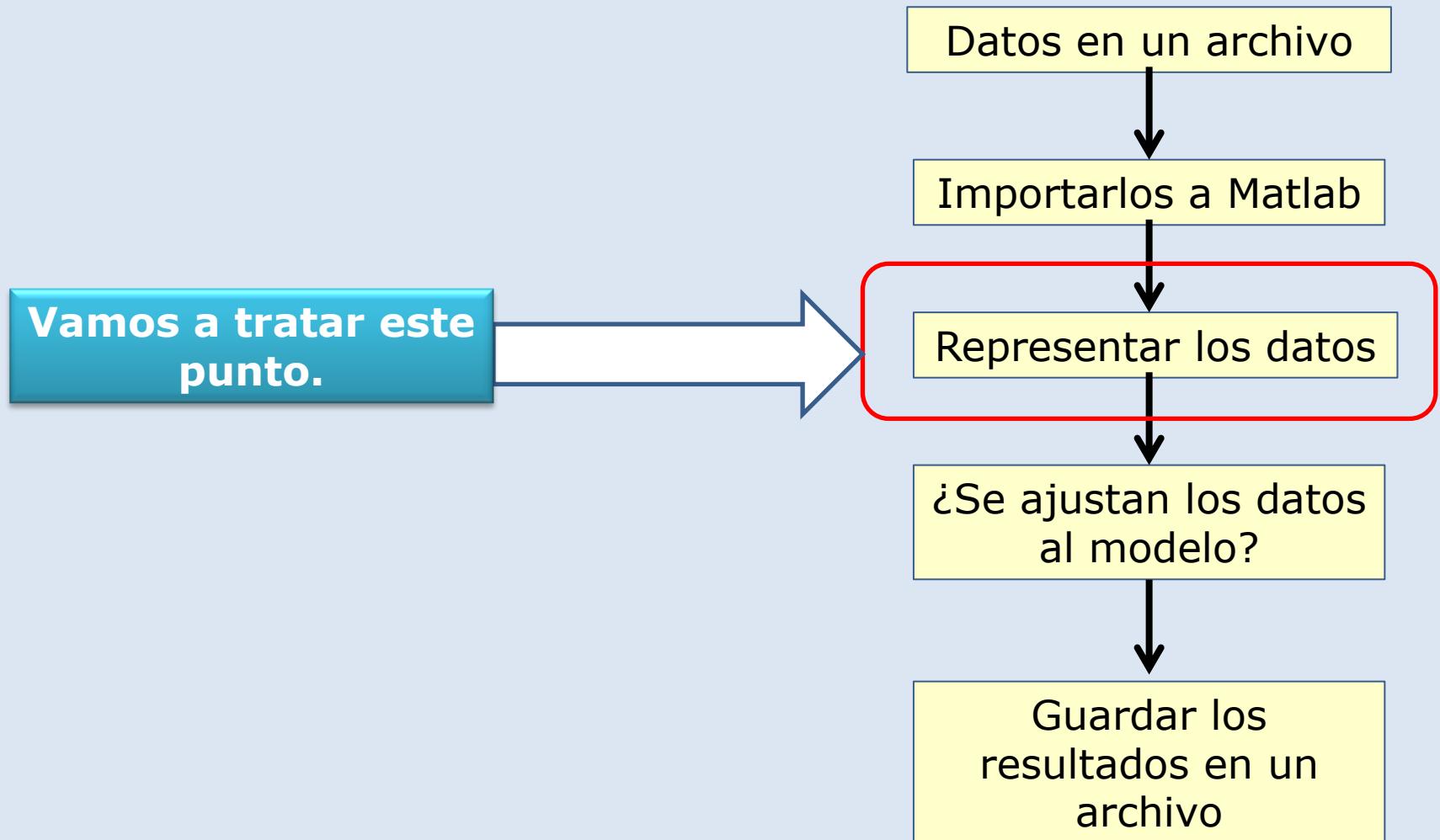
```
>> frase = sprintf('Una circunferencia tiene %5.3f %s \n',2*pi,unidades)

frase =

Una circunferencia tiene 6.283 radianes
```

Ejemplo de escritura de un archivo

Contenido del tema



Hacer gráficos con Matlab

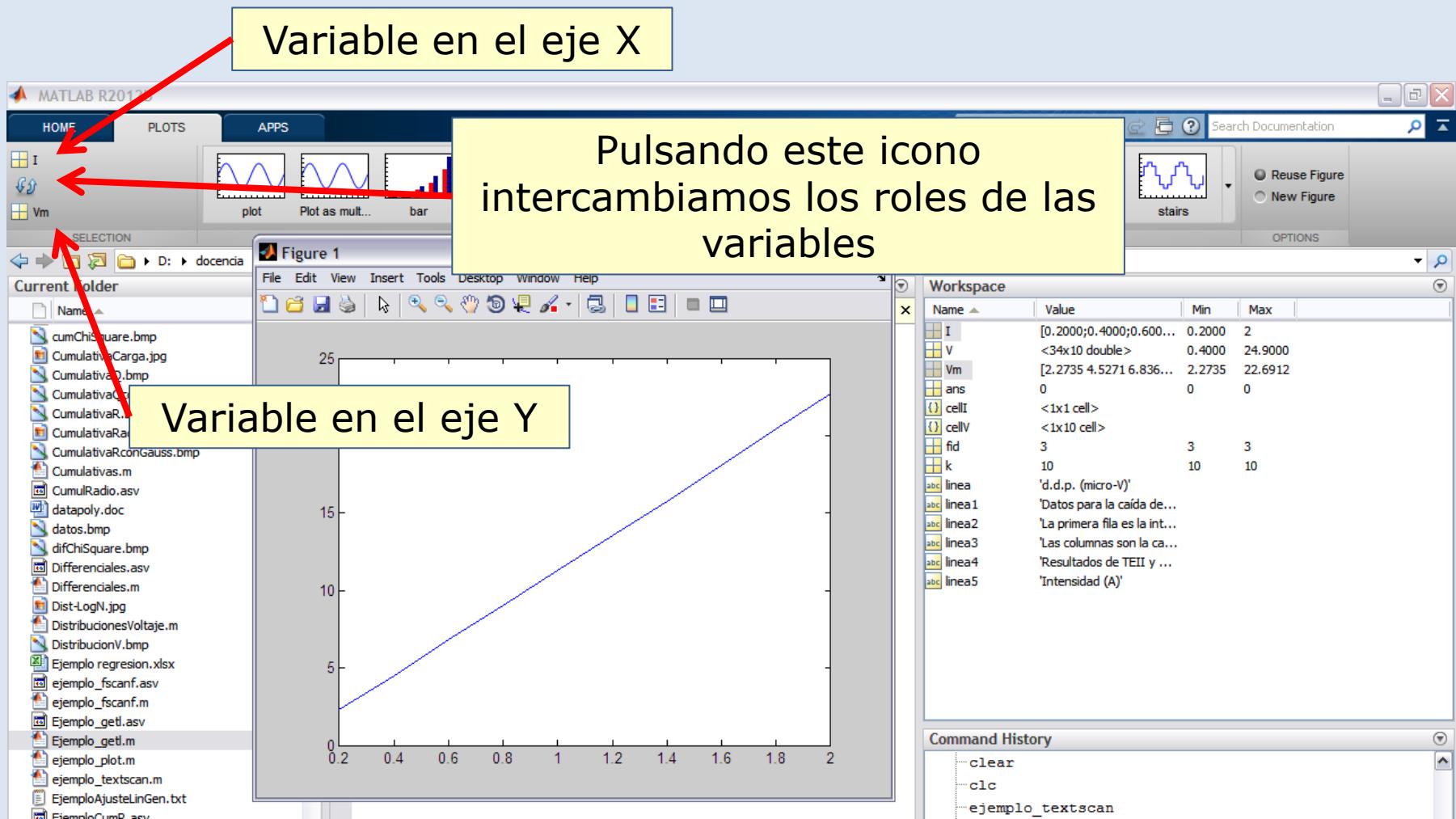
Una vez que tenemos variables en el **Workspace**, podemos hacer gráficos.

Pulsando con el botón izquierdo en el **Workspace** y yendo a la pestaña **Plot**, los gráficos posibles aparecen iluminados:



Gráficos Y frente a X

Es posible seleccionar más de una variable manteniendo pulsada la tecla **ctrl**. Hay que indicar a Matlab cuál es la variable a poner en el eje X y cuál en el eje Y



Gráficos 2D

Otros tipos de gráficos en dos dimensiones:

[plot](#)

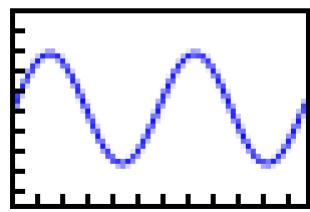


Gráfico con escalas lineales en ambos ejes.

[plotyy](#)

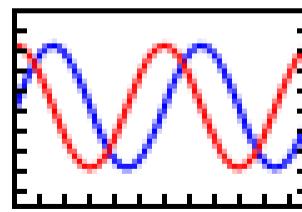


Gráfico con escalas lineales y dos ejes Y

[semilogx](#)

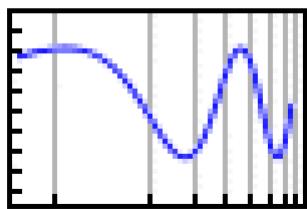


Gráfico con una escala Y lineal y X logarítmica

[semilogy](#)

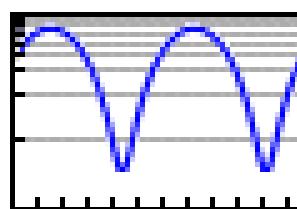


Gráfico con una escala X lineal e Y logarítmica

[loglog](#)

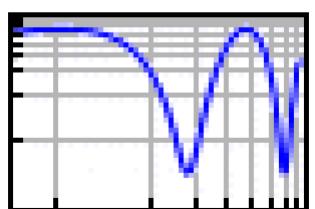
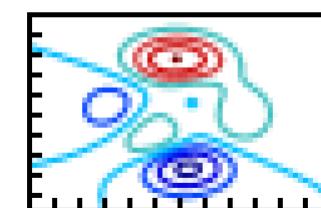


Gráfico con escala logarítmica en ambos ejes

[contour](#)



Representa curvas de nivel a partir del contenido de una matriz Z

Gráficos 3D

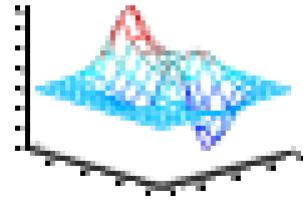
Otros tipos de gráficos en tres dimensiones:

plot3



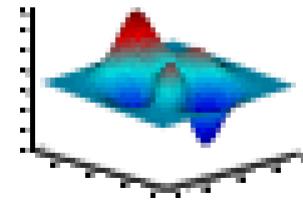
Representa una curva en tres dimensiones a partir de las coordenadas X, Y, Z de sus puntos (X, Y, Z son vectores)

mesh



Representa una superficie usando una red de líneas a partir de las coordenadas X Y Z de sus puntos. (X, Y, Z son matrices)

surf



Representa una superficie “sólida” a partir de las coordenadas X Y Z de sus puntos. (X, Y, Z son matrices)

Ventanas gráficas: función *figure*.

También es posible crear gráficos programáticamente. Para ello, hay que empezar creando la ventana que ocupará el gráfico con:

figure(N)

ó

id=figure(N)

Crea una ventana en la que es posible dibujar un gráfico.

N: número o variable numérica que indica el número de ventana gráfica.

id: identificador (opcional) para la figura.

Si la ventana gráfica ya existe, la convierte en la ventana activa.

Las instrucciones referentes a gráficos que veremos a continuación se aplican sobre la ventana activa.

Gráficos

La función más básica para crear un gráfico en una ventana gráfica es:

plot(x1,y1,'aspecto1',x2,y2,'aspecto2',...)

x1: vector con los valores de las abscisas se la serie 1

y1: vector con los valores de las ordenadas de la serie 2.

'aspecto1': cadena que controla la forma en que se representa la serie1.

Por defecto, plot reemplaza el contenido de la ventana activa.
Para evitar esto se usa el comando hold on (hold off)

Matlab tiene una gran cantidad de funciones gráficas. Todas se usan de un modo similar. Se recomienda que experimentéis por vuestra cuenta mirando el help de Matlab.

Estilos de línea y marcadores.

El tipo de línea y marcador y el color se indican en la entrada '*aspecto1*' en la forma

'*aspecto1*' = '*lineamarcadortcolor*'

linea

-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

marcador

+	Plus sign		
o	Circle	^	Upward-pointing triangle
*	Asterisk	v	Downward-pointing triangle
.	Point	>	Right-pointing triangle
x	Cross	<	Left-pointing triangle
'square' or s	Square	'pentagram' or p	Five-pointed star (pentagram)
'diamond' or d	Diamond	'hexagram' or h	Six-pointed star (hexagram)

color

r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Control de los ejes

En la ventana gráfica podemos editar manualmente muchas propiedades de los gráficos que creamos.

También existen comandos para editar propiedades de los gráficos automáticamente. Los más importantes son:

axis([xmin xmax ymin ymax])

Establece los límites de los ejes X e Y

***xlabel('Título eje X'),
ylabel('Título eje Y')***

Pone títulos a los ejes X e Y

title('Título del gráfico')

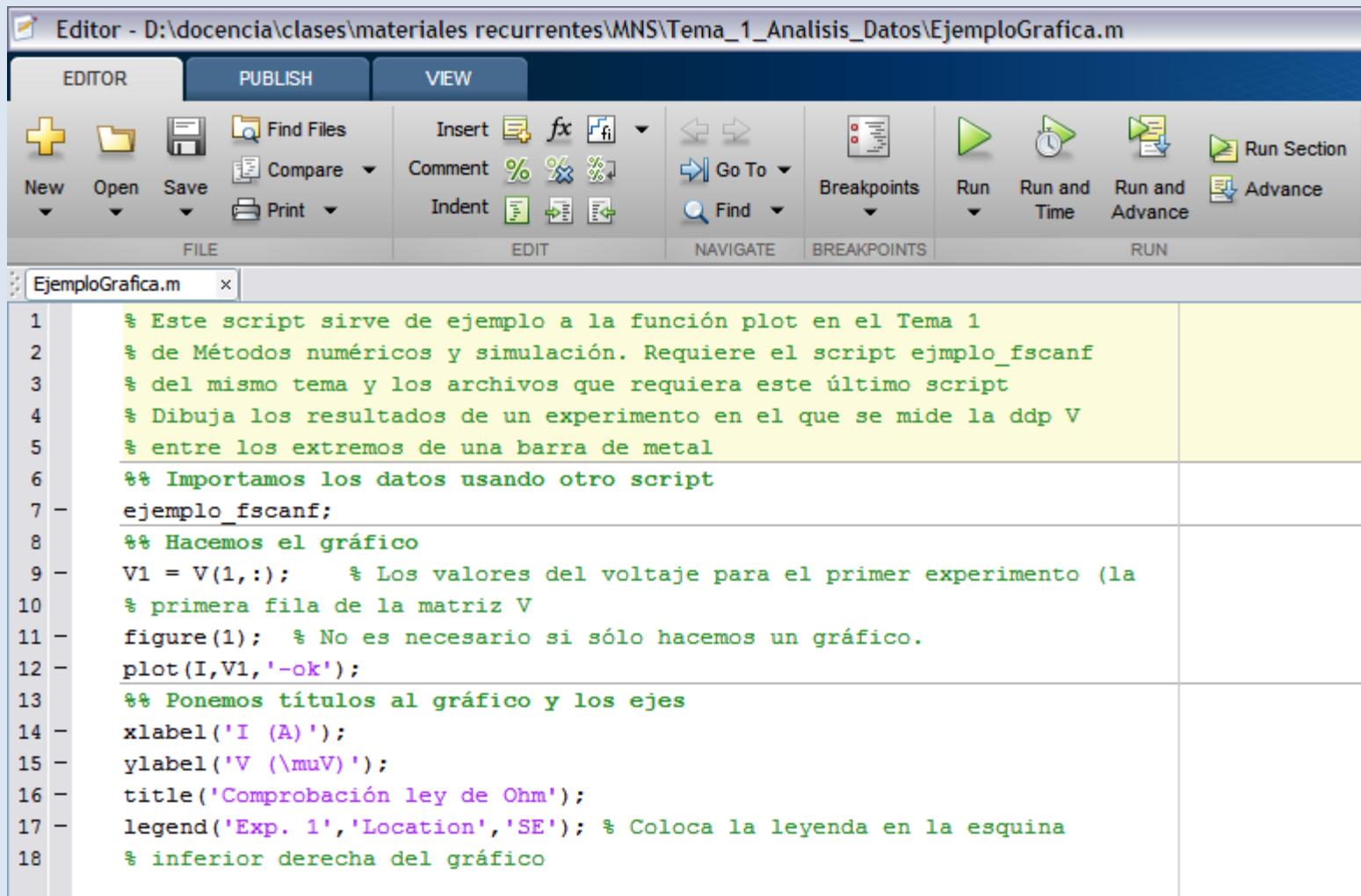
Pone un título al gráfico

legend('nombre1','nombre2',....)

Pone una leyenda con los nombres de las series de datos

Ejemplo.

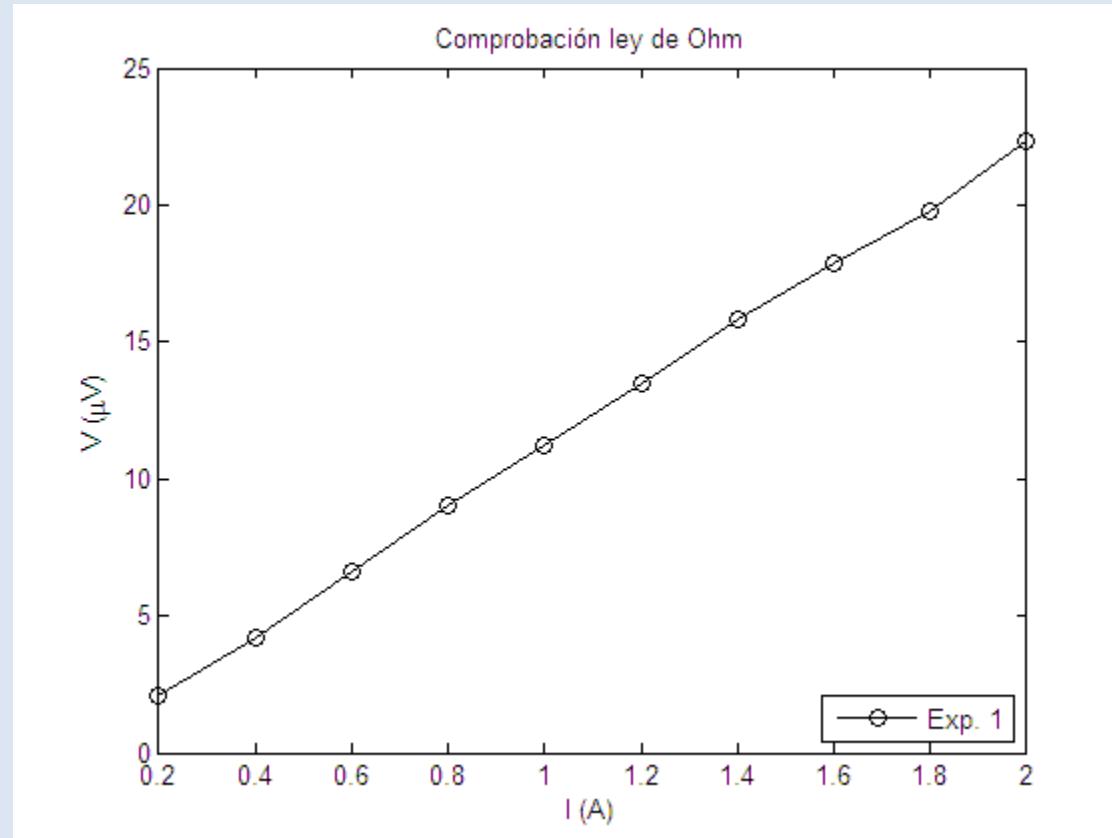
En este script se usan todos los comandos básicos para generar y editar gráficos.



```
EjemploGrafica.m % Este script sirve de ejemplo a la función plot en el Tema 1
% de Métodos numéricos y simulación. Requiere el script ejemplo_fscanf
% del mismo tema y los archivos que requiera este último script
% Dibuja los resultados de un experimento en el que se mide la ddp V
% entre los extremos de una barra de metal
%% Importamos los datos usando otro script
ejemplo_fscanf;
%% Hacemos el gráfico
V1 = V(1,:); % Los valores del voltaje para el primer experimento (la
% primera fila de la matriz V
figure(1); % No es necesario si sólo hacemos un gráfico.
plot(I,V1,'-ok');
%% Ponemos títulos al gráfico y los ejes
xlabel('I (A)');
ylabel('V (\mu V)');
title('Comprobación ley de Ohm');
legend('Exp. 1','Location','SE'); % Coloca la leyenda en la esquina
% inferior derecha del gráfico
```

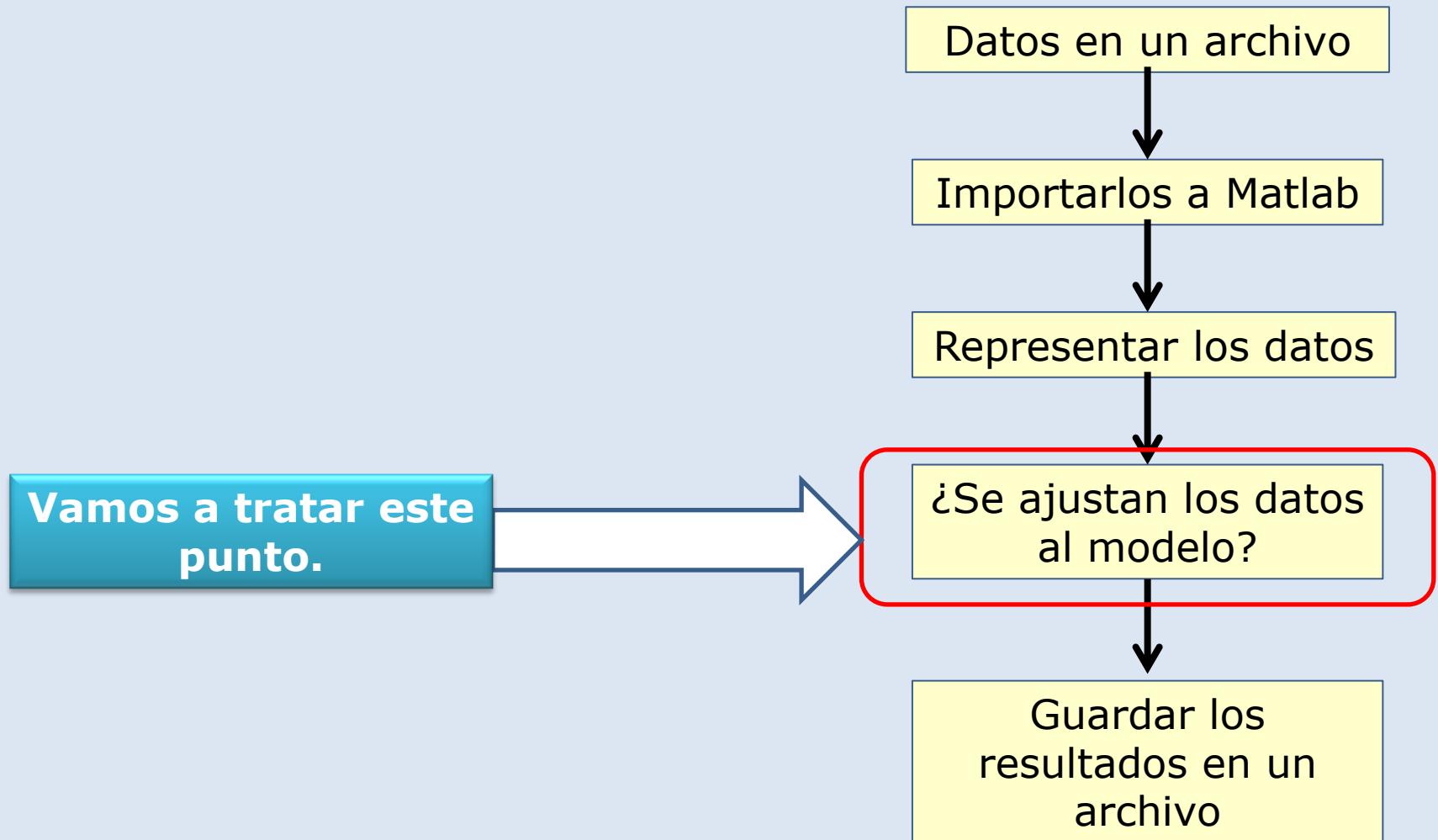
Ejemplo

Este es el gráfico que debe aparecer en pantalla:



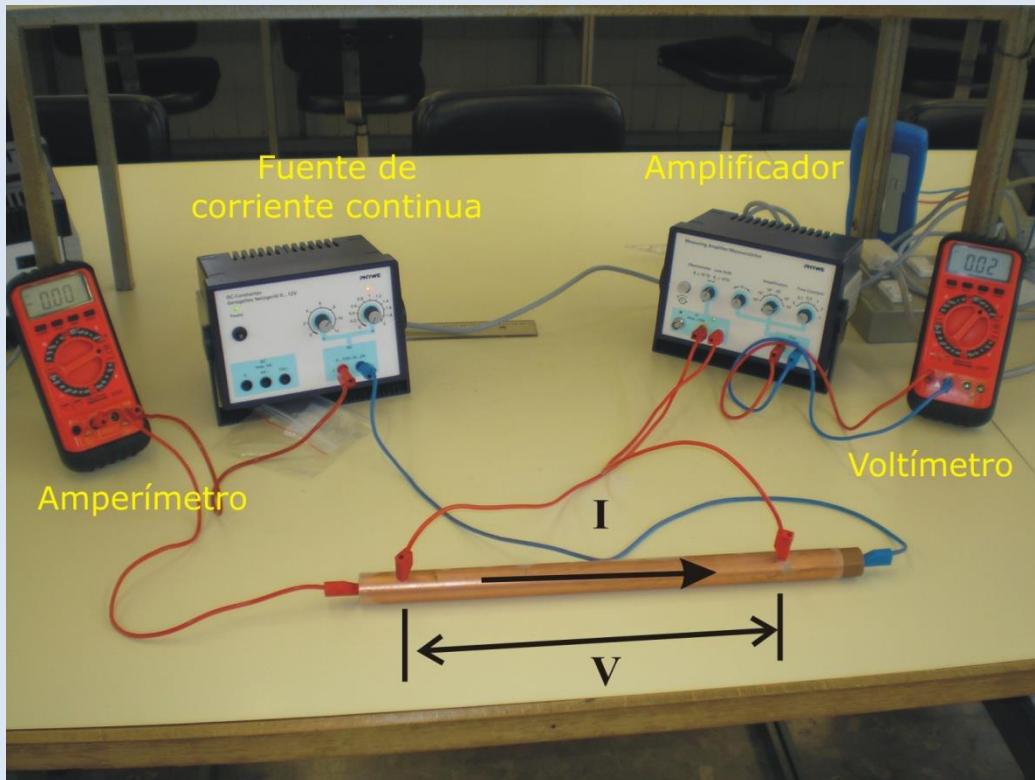
Vemos que los puntos experimentales se aproximan bastante a una recta.

Contenido del tema



Motivación

El archivo Tabla-datos.txt contiene los resultados de una práctica en la que se mide la diferencia de potencial V entre los extremos de una barra de cobre en función de la intensidad de corriente I que circula por la barra.



- La corriente I está impuesta por nosotros, es decir, **podemos elegir cuánto vale**. Diremos que es la variable **independiente**.
- La diferencia de potencial V no la podemos controlar, **sólo podemos medirla**. Diremos que es la variable **dependiente**.

Para que las próximas transparencias sean más generales, la variable independiente será la **X** y la dependiente la **Y**.

Tipos de ajustes lineales.

Cuando se hace un experimento, solemos tener un **modelo** $y=f(x; a_1, a_2, \dots, a_N)$ que describe como depende la variable Y de la variable X

a_1, a_2, \dots, a_N son **parámetros** (o **coeficientes**) del modelo cuyo valor queremos determinar.

- Cuando el modelo es de la forma:

$$f(x; a_1, a_2, \dots, a_N) = a_1 g_1(x) + a_2 g_2(x) + \dots + a_N g_N(x)$$

Se dice que tenemos que hacer un **ajuste lineal generalizado**.

- Si las $g_j(x)$ son de la forma x^{j-1} , entonces:

$$f(x; a_1, a_2, \dots, a_N) = a_1 + a_2 x + \dots + a_N x^{N-1}$$

Se dice que hacemos un **ajuste polinómico**.

Si $N=1$, el ajuste es el comúnmente llamado "lineal".

Aplicación a nuestro problema

En nuestro caso, el modelo es: $V = IR$

luego: $f(x; a_1, a_2) = a_2 x + a_1$

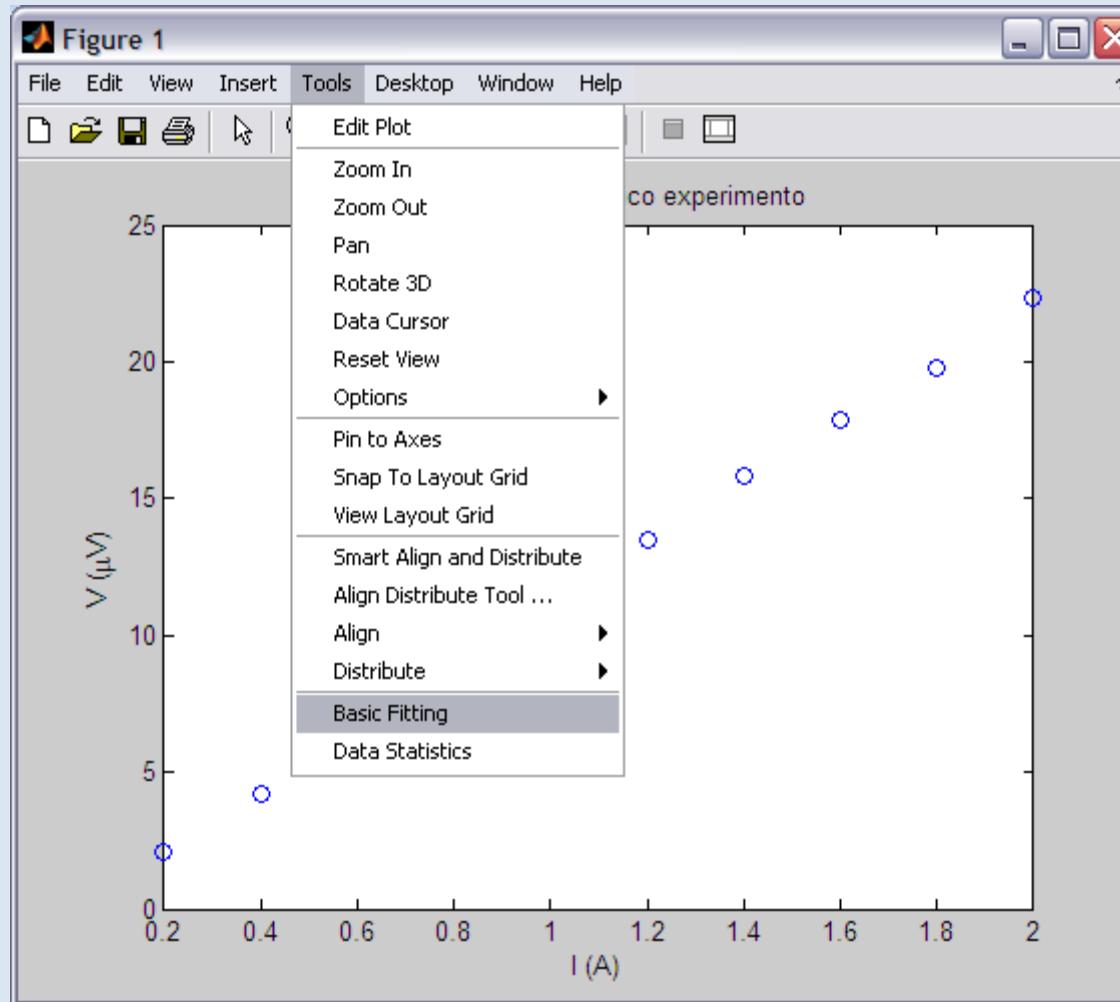
Y hacemos las identificaciones $x \rightarrow I$ $y \rightarrow V$ $a_2 \rightarrow R$

Esperamos que a_2 sea próximo al valor real R_{true} de la resistencia de la barra y que a_1 sea próximo a cero.

Tenemos que hacer un ajuste polinómico de grado 1 (comúnmente llamado "lineal")

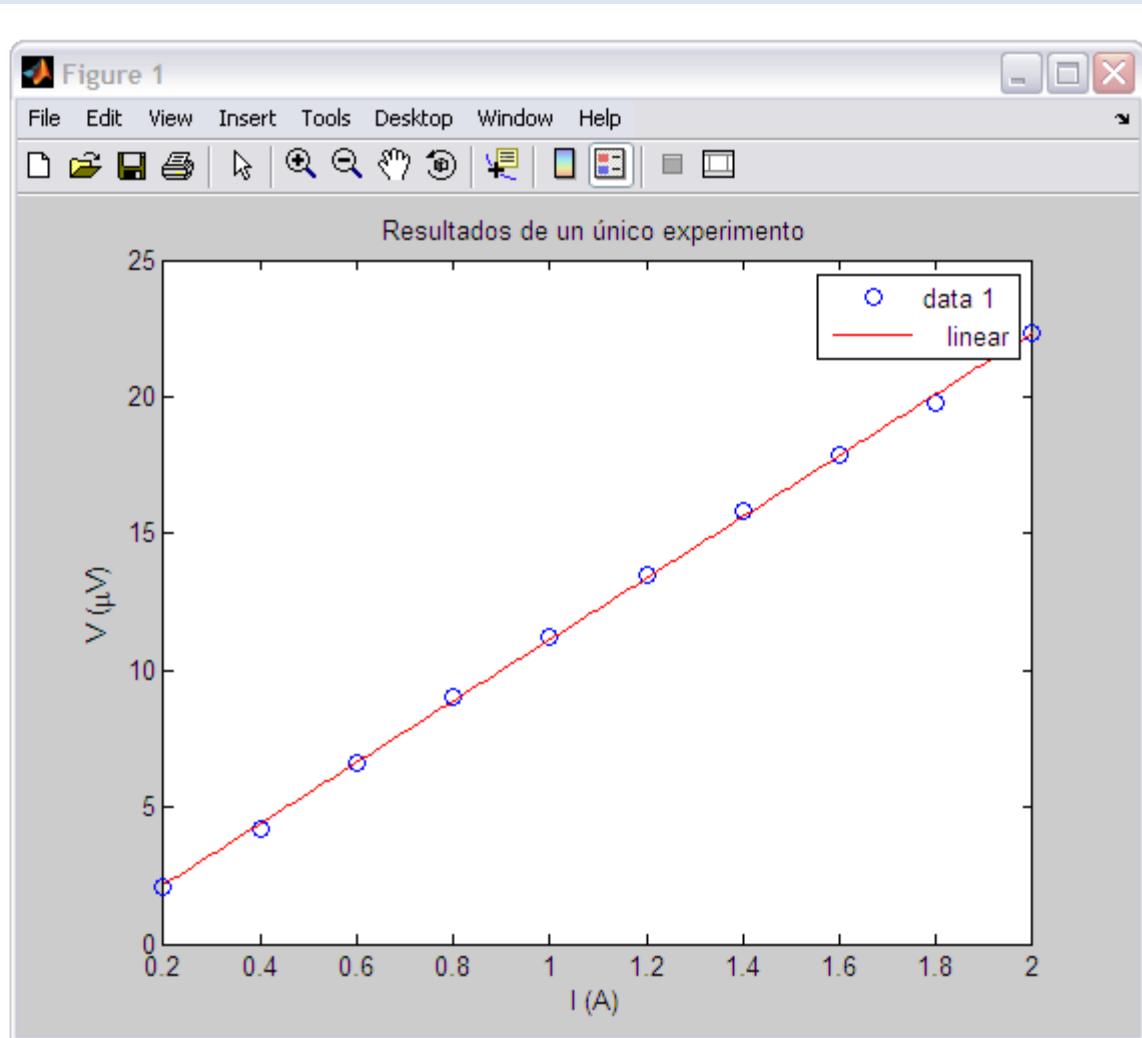
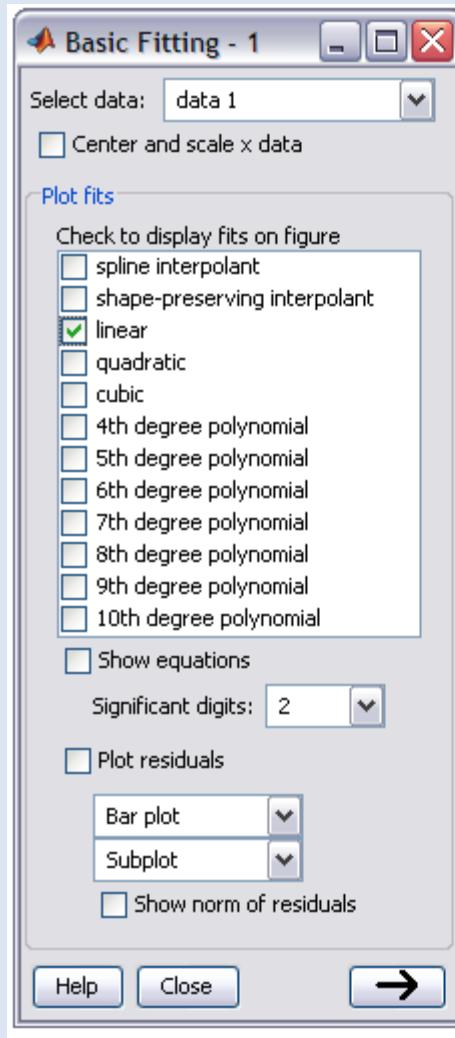
El menú “Basic Fitting”

Podemos ajustar una recta a los datos desde el menú *Tools/Basic Fitting*...



El menú “Basic fitting”

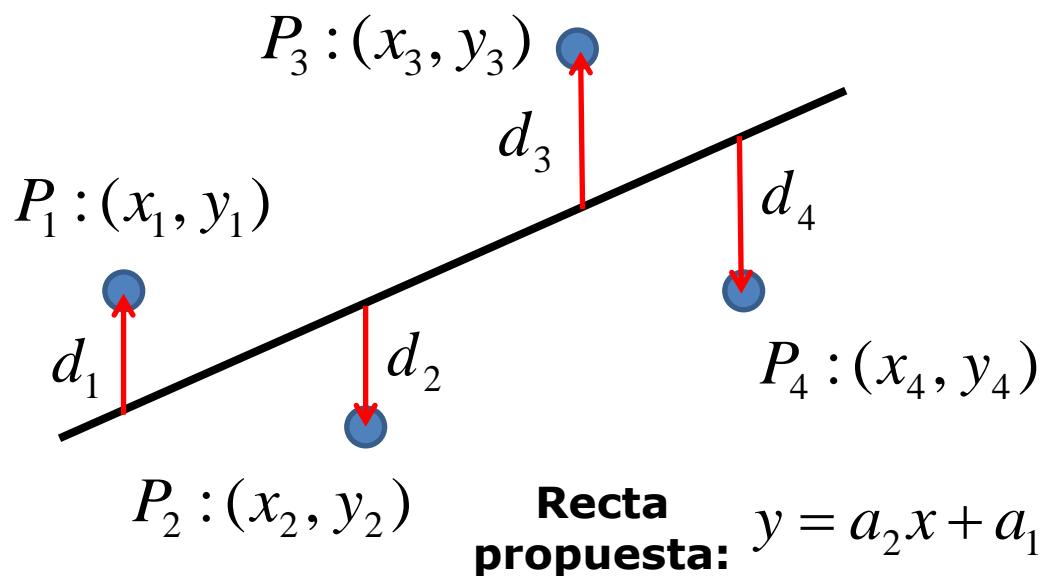
Y seleccionando la opción “linear”



Residuos

Hay que definir una función mérito que mida el acuerdo entre los datos experimentales y la función que queremos ajustar.

Como los puntos no están perfectamente alineados en una recta, si proponemos una recta cualquiera, habrá cierta distancia en vertical d_i entre cada punto i y la recta que hemos propuesto.



A las distancias d_i se las llama **residuos**.

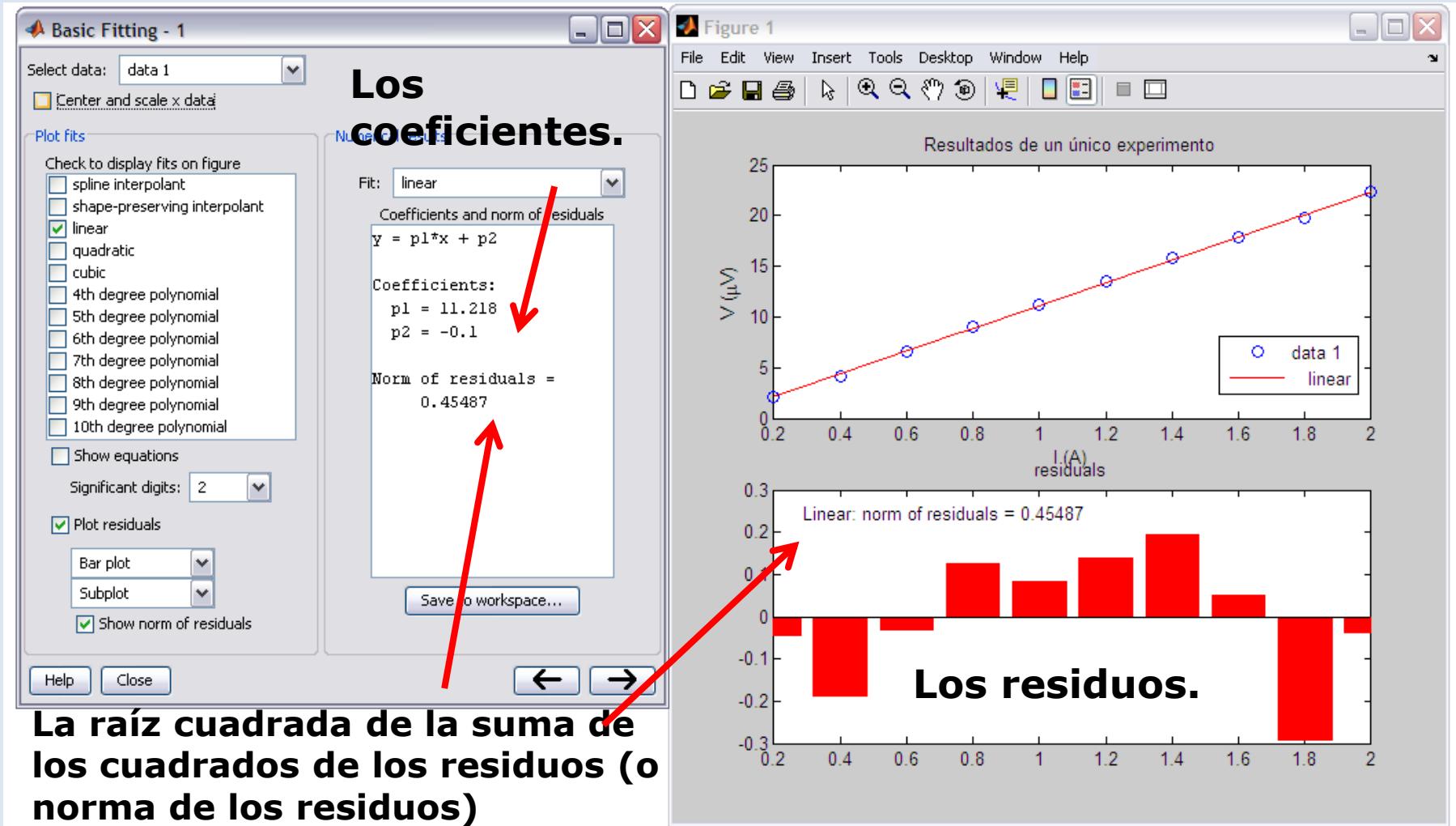
Por ejemplo, para d_3

$$d_3 = y_3 - (a_2x + a_1)$$

Los residuos pueden ser positivos o negativos.

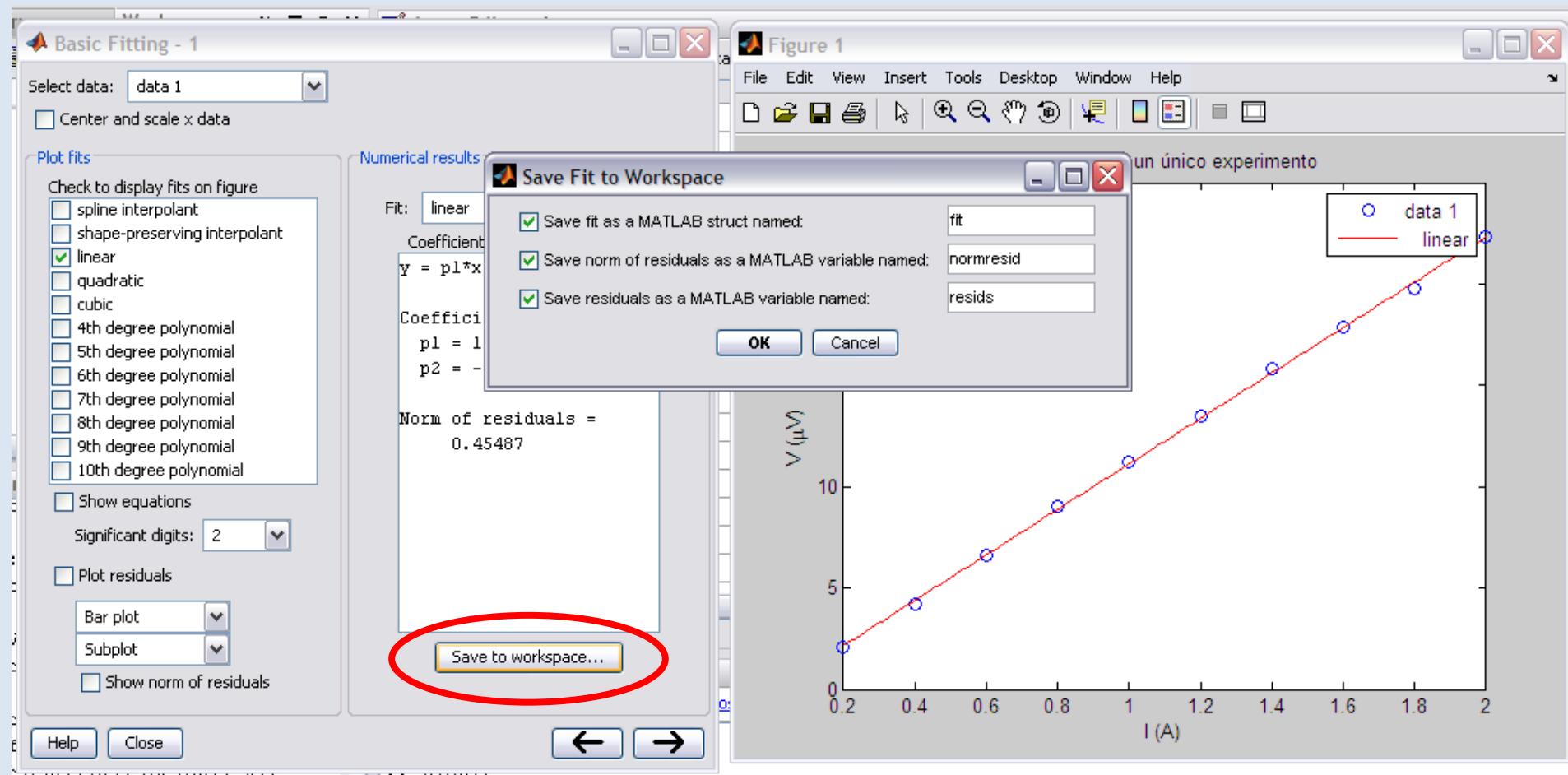
Visualización de los residuos

Una vez resueltas estas ecuaciones (que Matlab hace por nosotros), Matlab nos proporciona:

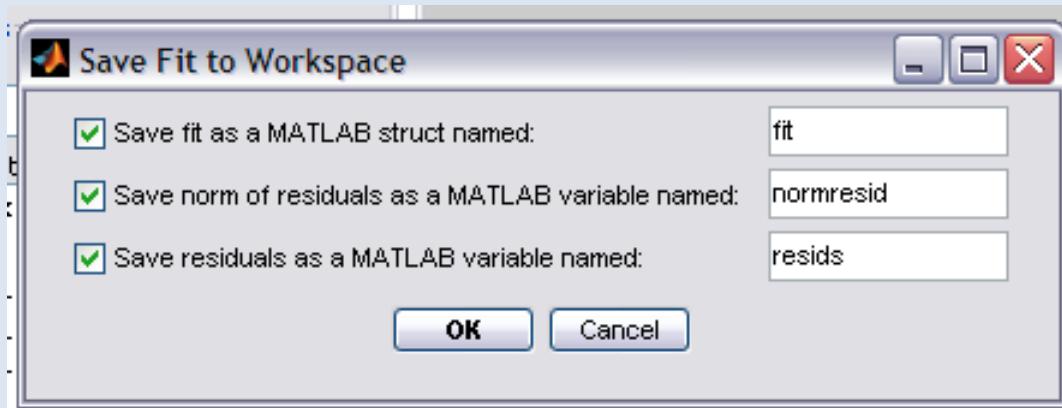


Exportar los resultados

Para grabar los resultados en el espacio de trabajo, se pica en el botón “Save to workspace” y se dan los nombres que queremos a las variables que se exportan.



¿Qué resultados se exportan?

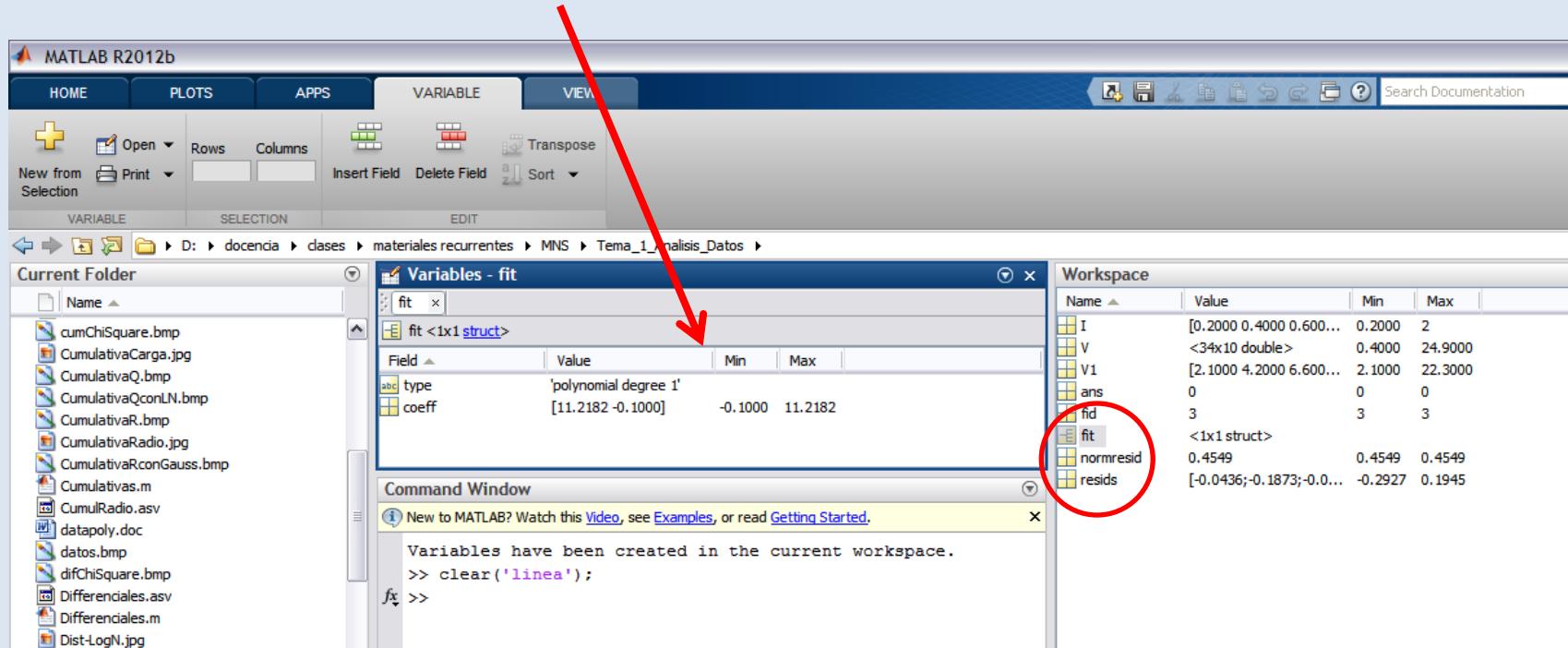


- Una estructura que contiene el tipo de ajuste y los coeficientes.
- Una variable con la norma de los residuos
- Un vector con los valores de los residuos.

Estas variables aparecen en el espacio de trabajo (workspace) de Matlab cuando picamos en OK.

Ejemplo

Aquí muestro los campos de la estructura fit. En el tema 0 vimos qué es una estructura.



La función *polyfit*

Si queremos hacer esta cuenta por línea de comando o en un programa, podemos usar la función ***polyfit***.

$$[p \ S] = \text{polyfit}(x, y, n)$$

Parámetros de entrada:

x: Vector con los valores de la variable independiente.

y: Vector con los valores de la variable dependiente.

n: orden del polinomio que se ajusta.

Parámetros de salida:

p: vector con los valores de los parámetros resultantes del ajuste.

S: Estructura que contiene otros resultados del ajuste.

Ejemplo: teclead $[p \ S] = \text{polyfit}(I, V1, 1)$

Ejemplo

Aquí muestro los campos de la estructura S que devuelve polyfit

The screenshot shows the MATLAB R2012b desktop environment. A red arrow points from the text above down to the 'Variables' pane.

Current Folder: Shows various files in the 'docencia > clases > materiales recurrentes > MNS > Tema_1_Analisis_Datos' directory.

Variables - S: Displays the contents of the struct variable 'S'.

Field	Value	Min	Max
R	$[-3.9243 \quad -2.8031; 0 \quad -1. \quad -3.9243 \quad 0]$		
df	8	8	8
normr	0.4549	0.4549	0.4549

Workspace: Shows the variables and their values in the workspace.

Name	Type	Value	Min	Max
I		[0.2000 0.4000 0.600...]	0.2000	2
S	struct	<1x1 struct>		
V		<34x10 double>	0.4000	24.9000
V1		[2.1000 4.2000 6.600...]	2.1000	22.3000
ans		0	0	0
fid		3	3	3
p		[11.2182 -0.1000]	-0.1000	11.2182

Command Window: Displays the command and its output.

```
>> [p S]=polyfit(I,V1,1)
p =
    11.2182   -0.1000

S =
    R: [2x2 double]
    df: 8
    normr: 0.4549
```

Command History: Shows the last command entered.

```
figure(id2)
```

Estructura devuelta por *polyfit*

Una **estructura** es un tipo especial de contenedor de datos que alberga varios **campos**.

Cada campo puede contener datos de una naturaleza diferente.

Su símbolo en Matlab es:



A cada campo se accede escribiendo: Nombre-Estructura.Campo

Por ejemplo, en el caso de la estructura S que devuelve polyfit

S.R da la matriz R que resulta de la descomposición QR de la matriz de diseño del problema.

S.normr da la norma de los residuos

S.df da el número de puntos menos el número de parámetros del ajuste (df= número de grados de libertad)

La función *polyval*

Supongamos que hemos hecho un ajuste **polinómico**:

$$f(x; a_1, a_2 \dots, a_N) = a_1 + a_2 x + \dots + a_N x^{N-1}$$

Y queremos usarlo para evaluar el valor y_o predicho para la variable y para un cierto valor x_o con su grado de incertidumbre. Para ello usamos la función:

[yo,delta] = polyval(p,xo,S)

Parámetros de entrada:

p: vector con los coeficientes del ajuste.

xo: valor de la variable independiente donde evaluamos el ajuste.

S: estructura devuelta por la función *polyfit*.

Parámetros de salida:

yo: valor predicho para la variable dependiente.

delta: intervalo de confianza para yo. El 50% de los valores experimentales para la variable independiente estará dentro de (yo-delta,yo+delta).

Fundamentos matemáticos.

**Valores óptimos de los
coeficientes del ajuste.**

**Incertidumbre de los
coeficientes del ajuste.**

**Determinación de la bondad
del ajuste.**

Valores óptimos de los coeficientes del ajuste

Consideremos un modelo:

$$f(x; a_1, a_2 \dots, a_N) = a_1 g_1(x) + a_2 g_2(x) + \dots + a_N g_N(x)$$

Encontrar los valores óptimos de los parámetros de ajuste a_1, a_2, \dots, a_N consiste en encontrar aquellos valores que sustituidos en a_1, a_2, \dots, a_N hacen que la curva $f(x)$ pase cerca de los puntos experimentales (x_j, y_j) .

Ajuste por mínimos cuadrados

La forma de encontrar los valores óptimos de los coeficientes es minimizar la suma de los cuadrados de los residuos. Por ejemplo, para un ajuste polinómico de grado 1:

$$Norma^2 = \sum_{i=1}^N d_i^2 = \sum_{i=1}^N (y_i - a_2 x_i - a_1)^2$$

Para hallar el mínimo con lápiz y papel tenemos que hacer:

$$\frac{\partial Norma^2}{\partial a_2} = 0 \rightarrow -2 \sum_{i=1}^N (y_i - a_2 x_i - a_1) x_i = 0$$

$$\frac{\partial Norma^2}{\partial a_1} = 0 \rightarrow -2 \sum_{i=1}^N (y_i - a_2 x_i - a_1) = 0$$

Ecuaciones normales

Nos queda:

$$\sum_{i=1}^N (y_i x_i - a_2 x_i x_i - a_1 x_i) = 0$$

$$\sum_{i=1}^N (y_i - a_2 x_i - a_1) = 0$$

A estas ecuaciones se las llama **ecuaciones normales** del ajuste. se suelen escribir de la forma.

$$S_{xy} = a_2 S_{xx} + a_1 S_x$$

$$S_{xx} = \sum_{i=1}^N x_i x_i \quad S_{xy} = \sum_{i=1}^N x_i y_i$$

$$S_y = a_2 S_x + a_1 S$$

$$S = \sum_{i=1}^N 1 = N \quad S_x = \sum_{i=1}^N x_i \quad S_y = \sum_{i=1}^N y_i$$

En forma matricial:

$$\begin{pmatrix} S_y \\ S_{xy} \end{pmatrix} = \begin{pmatrix} S & S_x \\ S_x & S_{xx} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad B = \mathbf{S} \cdot A$$

A: Vector con los valores de los coeficientes

Solución de las ec. normales

Y su solución es:

$$\Delta = SS_{xx} - (S_x)^2 \quad a_2 = \frac{SS_{xy} - S_x S_y}{\Delta} \quad a_1 = \frac{S_{xx} S_y - S_x S_{xy}}{\Delta}$$

Y en forma matricial:

$$A = \mathbf{S}^{-1} \cdot B = \mathbf{C} \cdot B \quad \mathbf{C} = \mathbf{S}^{-1}$$

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} S_y \\ S_{xy} \end{pmatrix}$$

IMPORTANTE: Ni la matriz \mathbf{S} ni la matriz \mathbf{C} dependen de y_1, y_2, \dots, y_N , porque en la matriz \mathbf{S} sólo aparecen sumas de potencias de x_1, x_2, \dots, x_N .

Solución seguida por Matlab

Volvamos al ajuste lineal. Las ecuaciones de la que lo obtuvimos son:

$$f(x; a_2, a_1) = a_2 x + a_1$$
$$0 = \sum_{i=1}^N (y_i - a_2 x_i - a_1) \quad 0 = \sum_{i=1}^N (y_i x_i - a_2 x_i x_i - a_1 x_i)$$

Podemos escribirlas como:

$$0 = \sum_{i=1}^N y_i \times 1 - \sum_{i=1}^N (a_2 x_i + a_1 \times 1) \quad 0 = \sum_{i=1}^N y_i x_i - \sum_{i=1}^N (a_2 x_i x_i - a_1 x_i)$$

Lo que he hecho ha sido simplemente separar por un lado las sumas que sólo contienen x's y las que contienen algún valor d las y's.

Ecuaciones en forma matricial

$$0 = \sum_{i=1}^N y_i \times 1 - \sum_{i=1}^N (a_2 x_i + a_1) \times 1$$

$$0 = (1 \quad 1 \quad \dots \quad 1) \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix} - (1 \quad 1 \quad \dots \quad 1) \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$0 = \sum_{i=1}^N y_i x_i - \sum_{i=1}^N (a_2 x_i - a_1) \times x_i$$

$$0 = (x_1 \quad x_2 \quad \dots \quad x_N) \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix} - (x_1 \quad x_2 \quad \dots \quad x_N) \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

La matriz de diseño

Y ambas ecuaciones se pueden combinar como:

$$0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \dots \\ 1 & x_N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$V^t(x_i) \cdot Y = V^t(x_i) \cdot V(x_i) \cdot A$$

V: matriz NxM llamada **matriz de diseño** o matriz de **Vandermonde** del ajuste.

M: número de parámetros. N: número de puntos.

Y: vector columna con los valores de la variable dependiente

A: vector columna con los parámetros del ajuste.

¿Por qué calcular de esta forma?

Supongamos que ahora quiero ajustar N pares de datos experimentales por un polinomio de segundo orden:

$$f(x; a_1, a_2, a_3) = a_3x^2 + a_2x + a_1$$

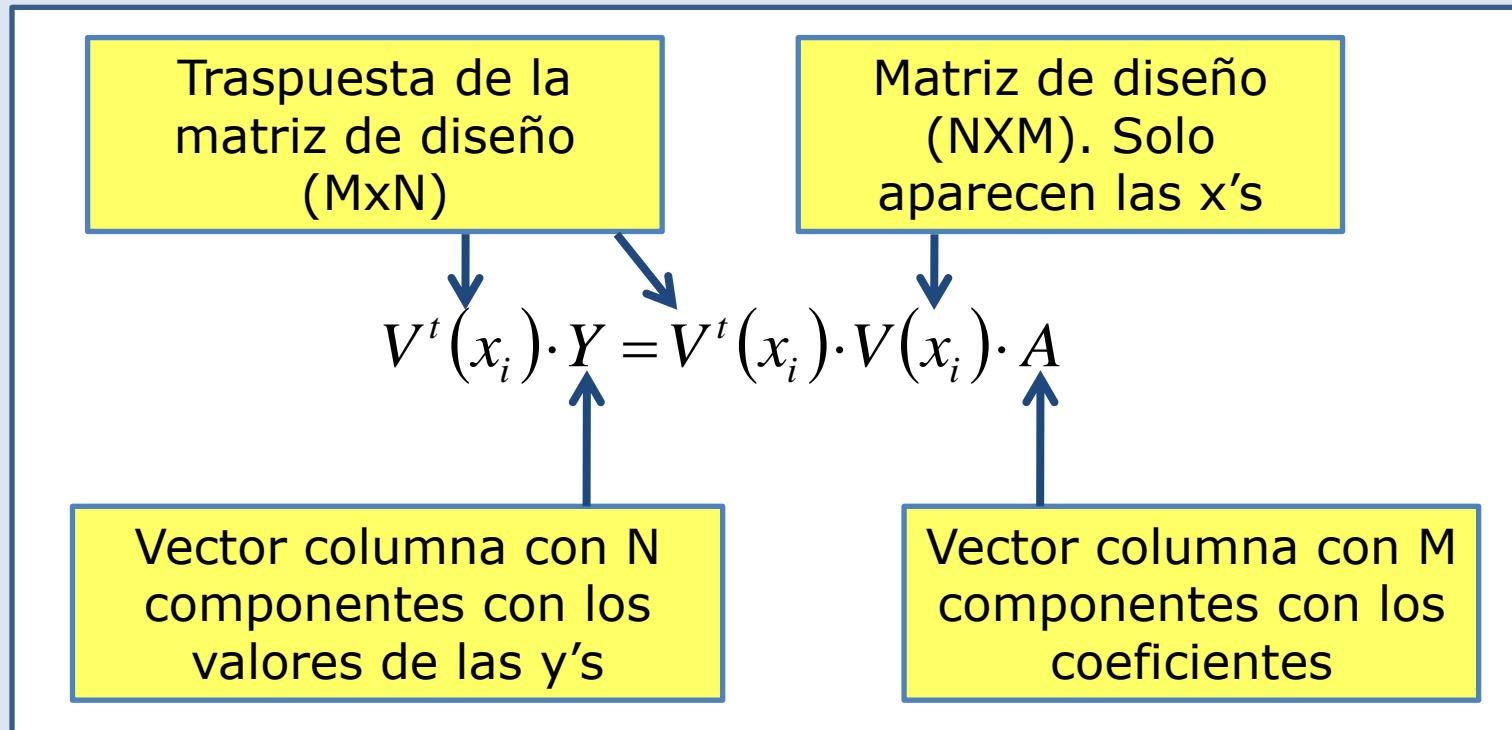
Repitiendo el procedimiento llego a un sistema de ecuaciones parecido

$$0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \\ x_1^2 & x_2^2 & \dots & x_N^2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \\ x_1^2 & x_2^2 & \dots & x_N^2 \end{pmatrix} \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \dots & \dots \\ 1 & x_N & x_N^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Puedo usar este procedimiento para polinomios de cualquier orden haciendo crecer la matriz de diseño hasta la potencia que desee.

Ajuste polinómico

En general, para ajustar por un polinomio de grado M-1:



N es el número de puntos

M es el número de parámetros.

El algoritmo de *polyfit* (I)

Este es el procedimiento que usa la función *polyfit(x,y,n)*

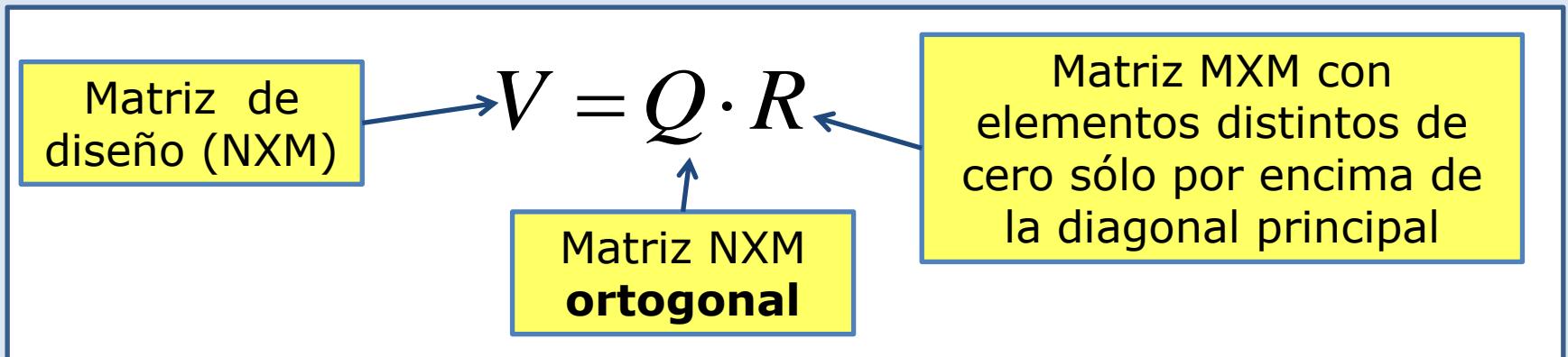
```
--  
52 - x = x(:);  
53 - y = y(:);  
54  
  
55 - if nargout > 2  
56 -     mu = [mean(x); std(x)];  
57 -     x = (x - mu(1))/mu(2);  
58 - end  
  
59  
60 % Construct Vandermonde matrix.  
61 - V(:,n+1) = ones(length(x),1,class(x));  
62 - for j = n:-1:1  
63 -     V(:,j) = x.*V(:,j+1);  
64 - end  
  
65  
66 % Solve least squares problem.  
67 - [Q,R] = qr(V,0);  
68 - ws = warning('off','all');  
69 - p = R\ (Q'*y); % Same as p = V\ y;
```

Escribe x,y como vectores columna

Construye la matriz de diseño

Descomposición Q-R

Siempre se puede descomponer la matriz de diseño en la forma:



Que la matriz Q sea ortogonal significa que:

$$Q^t \cdot Q = 1$$

La función de Matlab que hace esta descomposición es

$$[Q \ R] = qr(V, 0)$$

Esta es la matriz R que aparece en la estructura S que devuelve polyfit.

Obtención de los parámetros

La ecuación que tenemos que resolver es

$$V^t \cdot Y = V^t \cdot V \cdot A$$
$$(R^t \cdot Q^t) \cdot Y = (R^t \cdot Q^t) \cdot (Q \cdot R) \cdot A = R^t \cdot R \cdot A$$

Sacando factor común R^t :

$$R^t \cdot (Q^t \cdot Y - R \cdot A) = 0 \iff Q^t \cdot Y - R \cdot A = 0$$

Solución:

$$A = R^{-1} \cdot Q^t \cdot Y$$

A: Vector columna con M componentes que son los valores de los coeficientes

El algoritmo de *polyfit* (II)

Este es el procedimiento que usa la función *polyfit(x,y,n)*

```
--  
52 - x = x(:);  
53 - y = y(:);  
54  
  
55 - if nargout > 2  
56 -     mu = [mean(x); std(x)];  
57 -     x = (x - mu(1))/mu(2);  
58 - end  
  
59  
60 % Construct Vandermonde matrix.  
61 - V(:,n+1) = ones(length(x),1,class(x));  
62 - for j = n:-1:1  
63 -     V(:,j) = x.*V(:,j+1);  
64 - end  
  
65  
66 % Solve least squares problem.  
67 - [Q,R] = qr(V,0);  
68 - ws = warning('off','all');  
69 - p = R\ (Q'*y); % Same as p = V\ y;
```

Escribe x,y como vectores columna

Construye la matriz de diseño

Resuelve las ecuaciones.

Ajuste lineal generalizado

Supongamos que ahora quiero ajustar los datos por

$$f(x; a_1, a_2) = a_1 g(x) + a_2 h(x)$$

Donde $g(x)$, $h(x)$ y $w(x)$ son funciones conocidas que dependen exclusivamente de x .

Repetiendo el procedimiento llego al sistema de ecuaciones:

$$0 = \begin{pmatrix} g(x_1) & g(x_2) & \dots & g(x_N) \\ h(x_1) & h(x_2) & \dots & h(x_N) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} g(x_1) & g(x_2) & \dots & g(x_N) \\ h(x_1) & h(x_2) & \dots & h(x_N) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$
$$V^t(x_i) \cdot Y = V^t(x_i) \cdot V(x_i) \cdot A$$

Esta ecuación algebraica tiene la misma forma que hemos visto antes, luego los coeficientes del ajuste se calculan por el mismo método.

Procedimiento para ajuste lineal generalizado

1) Construir la matriz de diseño:

Suponiendo que los valores de la variable independiente están en un vector x

2) Hacer su descomposición QR:

3) Despejar los coeficientes:

$$V = \begin{pmatrix} g(x_1) & h(x_1) \\ g(x_2) & h(x_2) \\ \dots & \dots \\ g(x_N) & h(x_N) \end{pmatrix}$$

$$V = [g(x) \ h(x)]$$

$$[Q \ R] = qr(V, 0)$$

$$p = R \setminus Q' * y$$

Matlab no trae una función para hacer ajustes lineales generalizados.
Las cuentas las tenemos que hacer nosotros.

Incertidumbre de los coeficientes del ajuste.

Consideremos un modelo:

$$f(x; a_1, a_2 \dots, a_N) = a_1 g_1(x) + a_2 g_2(x) + \dots + a_N g_N(x)$$

Y dos conjuntos de datos experimentales (x_j, y_j) y (x_k, y_k) .

- Si ajustamos $f(x)$ a los puntos (x_j, y_j) los valores óptimos de los parámetros serán $a_1^{(1)}, a_2^{(1)}, \dots, a_N^{(1)}$.
- Si ajustamos $f(x)$ a los puntos (x_k, y_k) los valores óptimos de los parámetros serán $a_1^{(2)}, a_2^{(2)}, \dots, a_N^{(2)}$ (diferentes de $a_1^{(1)}, a_2^{(1)}, \dots, a_N^{(1)}$).

Dar la incertidumbre de los coeficientes del ajuste es dar el **rango en el que uno espera que varíen los valores óptimos** de los coeficientes de un conjunto de datos experimentales a otro.

Incertidumbre en las variables experimentales.

Normalmente, en un experimento:

La variable **independiente** se conoce **sin ningún tipo de incertidumbre**.

Esto significa que los valores x_1, x_2, \dots, x_M que se imponen a la variable independiente se pueden determinar con una **precisión muy grande**.

La variable **dependiente** tiene cierto nivel de **incertidumbre**.

Esto significa que si se mantiene el valor de la variable independiente en un número k medidas (por ejemplo el valor x_j), se obtienen k valores diferentes $y_{j1}, y_{j2}, \dots, y_{jk}$ de la variable dependiente.

Algunas aclaraciones previas

De momento, vamos a suponer que conocemos la incertidumbre de la variable dependiente, que llamaremos σ_y .

A la incertidumbre del coeficiente del ajuste a_j la llamaremos $\sigma(a_j)$

En las transparencias que siguen, explicaremos cómo calcular $\sigma(a_j)$ a partir de los datos experimentales (x_k, y_k) y la incertidumbre de la variable independiente σ_y .

Más adelante, explicaremos cómo conocer σ_y .

La matriz de covariancias

Retomamos el ajuste lineal:

$$f(x; a_2, a_1) = a_2 x + a_1$$

Sabemos que usando las ecuaciones normales a_1 y a_2 se encuentran de:

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} S_y \\ S_{xy} \end{pmatrix}$$

Sabemos la incertidumbre de la variable dependiente:

$$\sigma(y)$$

Se llama **matriz de covariancias** a la matriz:

$$\sigma(y)^2 \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1M} \\ C_{21} & C_{22} & \dots & C_{2M} \\ \dots & & & \dots \\ C_{M1} & C_{M2} & \dots & C_{MM} \end{pmatrix}$$

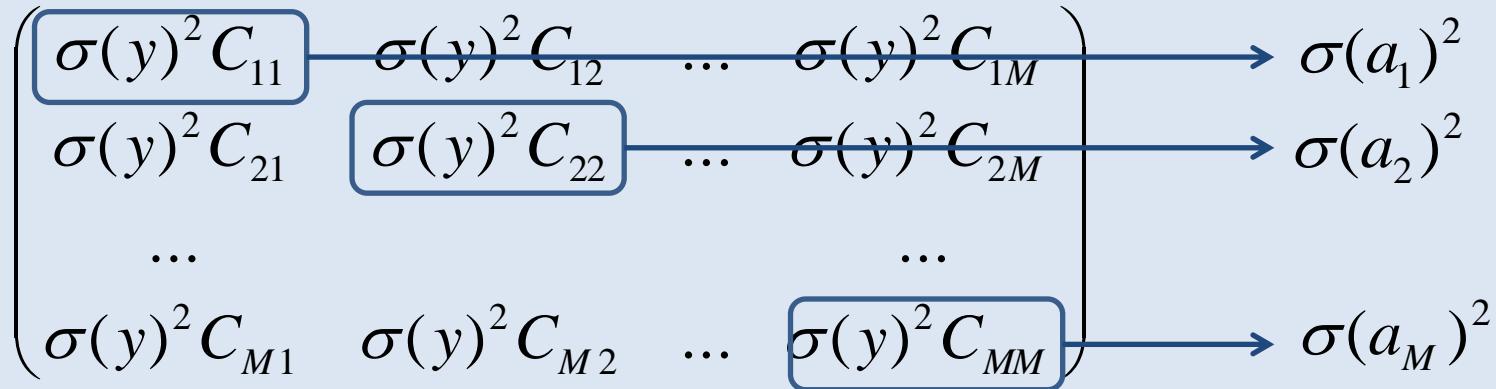
Incertidumbre en los parámetros

En general, si hacemos un ajuste del tipo:

$$f(x; a_1, a_2, \dots, a_M) = a_M x^{M-1} + \dots + a_2 x + a_1$$

Las incertidumbres en los parámetros del ajuste vienen dados por los elementos de la diagonal de la matriz de covariancias:

$$\sigma(a_j)^2 = \sigma(y)^2 C_{jj}$$



Pero polyfit(x,y,n) nos devuelve la matriz R. ¿Qué hacemos para hallar C?

Demo: Incertidumbre en a_1 (I)

Puesto que:

$$a_1 = C_{11}S_y + C_{12}S_{xy}$$

Si consideremos que a_1 es función de y_1, y_2, \dots, y_N , la fórmula de propagación de errores nos dice:

$$\sigma(a_1)^2 = \sum_{j=1}^N \sigma(y_j)^2 \left[\frac{\partial}{\partial y_j} (C_{11}S_y + C_{12}S_{xy}) \right]^2$$

$$\sigma(a_1)^2 = \sum_{j=1}^N \sigma(y_j)^2 \left[C_{11} \frac{\partial S_y}{\partial y_j} + C_{12} \frac{\partial S_{xy}}{\partial y_j} \right]^2$$

Puesto que C_{11} y C_{12} sólo dependen de x_1, x_2, \dots, x_N , y

$$\frac{\partial S_y}{\partial y_j} = \frac{\partial}{\partial y_j} \sum_{i=1}^N y_i = 1$$

$$\frac{\partial S_{xy}}{\partial y_j} = \frac{\partial}{\partial y_j} \sum_{i=1}^N x_i y_i = x_j$$

Demo: Incertidumbre en a_1 (II)

Sustituyendo llegamos a:

$$\sigma(a_1)^2 = \sum_{j=1}^N \sigma(y_i)^2 [C_{11} \times 1 + C_{12} x_j]^2$$

Y como todas las $\sigma(y_j)$ son iguales a $\sigma(y)$

Desarrollando el cuadrado.

$$\sigma(a_1)^2 = \sigma(y)^2 \sum_{j=1}^N [(C_{11})^2 \times 1 + (C_{12})^2 x_j^2 + 2C_{11}C_{12}x_j]$$

Sacando C_{11} y C_{12} factores comunes

$$\sigma(a_1)^2 = \sigma(y)^2 [(C_{11})^2 S + (C_{12})^2 S_{xx} + 2C_{11}C_{12}S_x]$$

Reordenando: $\sigma(a_1)^2 = \sigma(y)^2 [C_{11}(C_{11}S + C_{12}S_x) + C_{12}(C_{12}S_{xx} + C_{11}S_x)]$

Usando que $C = S^{-1}$ tenemos el resultado:

$$\sigma(a_1)^2 = \sigma(y)^2 C_{11}$$

Cálculo de la matriz de covariancias

Comparando los dos procedimientos:

Ecuaciones normales.

$$\begin{pmatrix} S_y \\ S_{xy} \end{pmatrix} = \begin{pmatrix} S & S_x \\ S_x & S_{xx} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad B = \mathbf{S} \cdot A$$

$$0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix} - \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_N \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$V^t(x_i) \cdot Y = V^t(x_i) \cdot V(x_i) \cdot A$$

Conclusión:

$$B = V^t \cdot Y \quad \mathbf{S} = V^t \cdot V$$

Cálculo de la matriz de covariancias

Como sabemos que:

$$\mathbf{S} = \mathbf{V}^t \cdot \mathbf{V} \quad \mathbf{V} = \mathbf{Q} \cdot \mathbf{R}$$

Sustituyendo:

$$\mathbf{S} = \mathbf{R}^t \cdot \mathbf{Q} \cdot \mathbf{Q}^t \cdot \mathbf{R} = \mathbf{R}^t \cdot \mathbf{R}$$

$$C = \mathbf{S}^{-1} = (\mathbf{R}^t \cdot \mathbf{R})^{-1} = \mathbf{R}^{-1} \cdot (\mathbf{R}^{-1})^t$$

Y de esta forma encontramos la matriz de covarianzas. Por eso polyfit(x,y,n) sólo devuelve la matriz R.

Recuerda que la inversa T de una matriz R se calcula como
 $T=inv(R)$.

Cuantificación de la incertidumbre σ_y de la variable dependiente.

Caso 1. Para cada valor x_j de la variable independiente hemos medido ***k* valores** $y_{j1}, y_{j2}, \dots, y_{jk}$ de la variable dependiente y hemos encontrado que tienen cierta dispersión.

Solución: Podemos entonces cuantificar la incertidumbre σ_y de la variable dependiente usando la desviación estándar s_y de los valores $y_{j1}, y_{j2}, \dots, y_{jk}$.

$$\sigma_y = \frac{s_y}{\sqrt{k}} \quad s_y = \sqrt{\frac{1}{N} \sum_{j=1}^k (y_j - \bar{y})^2}$$

s_y : desviación estándar de $y_{j1}, y_{j2}, \dots, y_{jk}$

k : no. de valores en $y_{j1}, y_{j2}, \dots, y_{jk}$

La función de Matlab:

B=std(A)

Calcula la desviación estándar de los valores contenidos en el array A. Si A es una matriz, B contiene la desviación estándar de los elementos de cada columna de A.

Cuantificación de la incertidumbre σ_y de la variable dependiente.

Caso 2. Conocemos el valor de σ_y por otros medios; p. ej., porque estimamos que es igual a la precisión del aparato de medida.

Para que esto ocurra, la variabilidad en los valores de la variable dependiente y (manteniendo el valor de la variable independiente x constante) debe ser menor que la precisión del aparato de medida.

Solución: Usar directamente el valor de σ_y para encontrar la matriz de covarianzas.

Cuantificación de la incertidumbre σ_y de la variable dependiente.

Caso 3. No conocemos el valor de la incertidumbre σ_y de la variable dependiente.

Típicamente esto ocurre cuando para cada valor x_j de la variable independiente hayamos medido sólo un valor y_j de la variable dependiente y al variabilidad de los valores de la variable dependiente y es mayor que la precisión del aparato que usamos para medirla.

Solución: Estimar el valor de σ_y de la forma:

$$\sigma_y = \sqrt{\frac{Norma^2}{N - M}}$$

Norma: Norma de los residuos

M: número de parámetros. **N:** número de puntos experimentales.

N-M: número de grados de libertad.

Determinación de la bondad del ajuste.

Consideremos un modelo:

$$f(x; a_1, a_2 \dots, a_N) = a_1 g_1(x) + a_2 g_2(x) + \dots + a_N g_N(x)$$

Y un conjunto de datos experimentales (x_j, y_j) .

Determinar la bondad del ajuste de la curva $f(x)$ a los puntos (x_j, y_j) significa dar alguna medida de **cómo de cerca** pasa la curva $f(x)$ de los puntos experimentales (x_j, y_j) .

Distinguiremos entre el caso de un ajuste polinómico de grado 1 y el resto de ajustes.

El coeficiente de correlación

Para un **ajuste polinómico de grado 1** hay una forma sencilla de evaluar cómo de bueno es el ajuste. Para ello se define el coeficiente de correlación r (también llamado coeficiente de Pearson) como:

$$r = \frac{\sum_j (x_j - \langle x \rangle)(y_j - \langle y \rangle)}{\sqrt{\sum_j (x_j - \langle x \rangle)^2} \sqrt{\sum_j (y_j - \langle y \rangle)^2}}$$

Si $r=+/-1$, los datos se ajustan a una recta.

Si r es próximo a cero los datos no se ajustan a una recta.

Cálculo de r

En Matlab se calcula mediante la función

$Cr=corrcoef(x,y)$

Parámetros de entrada: un vector **x** con los valores de la variable independiente y un vector **y** con los valores de la variable dependiente.

Parámetros de salida: una matriz Cr con la forma:

$$Cr = \begin{pmatrix} 1 & r \\ r & 1 \end{pmatrix}$$

r es el coeficiente de correlación

Ejemplo: escribid corrcoef(I,V1)

Estimación de la bondad del ajuste

Si conocemos σ_y , (casos 1 y 2) podemos estimar la bondad del ajuste mediante la siguiente regla:

Si

$$\sigma_y \approx \sqrt{\frac{Norma^2}{M - N}}$$

El ajuste es bueno.

Si

$$\sigma_y \ll \sqrt{\frac{Norma^2}{M - N}}$$

La función $f(x)$ que hemos escogido no ajusta bien los datos experimentales (es decir, no pasa cerca de los puntos experimentales)

Si

$$\sigma_y \gg \sqrt{\frac{Norma^2}{M - N}}$$

La función $f(x)$ que hemos escogido ajusta “demasiado bien” los datos experimentales. O bien hemos sobre-estimado σ_y o hay algo “raro” en los datos.

Bibliografía del tema

Si quieres saber más sobre los contenidos del tema, lee en:

- Importación y exportación de datos
 - Capítulo 6 del Aprenda Matlab 7.0 como si estuviera en primero.
 - Capítulos 2.9 y 4 del *Essential Matlab for Engineers and Scientists*.
- Gráficos
 - Capítulos 8 y 9 del Aprenda Matlab 7.0 como si estuviera en primero.
 - Capítulo 7 del *Essential Matlab for Engineers and Scientists*.
- Ajuste de datos generalizado usando el método de los mínimos cuadrados.
 - Capítulos 2 y 15 del Numerical Recipes in Fortran 77
 - Capítulo 5 de Numerical Computing with Matlab. Cleve Moler.