Computación 1
- 2008 -
Manipulación de archivos

Necesidades

- Guardar en archivos datos del espacio de trabajo.
- Recuperar datos guardados previamente.

Guardar datos

- Copiar (al portapapeles) desde la Ventana de Comandos de Matlab y pegar en un archivo de texto.
- Guardar datos en un archivo binario o ASCII usando la función `save`.
- Guardar planillas de cálculo, datos científicos, imágenes o audio con funciones que vienen con las herramientas de Matlab.
- Guardar datos en un archivo utilizando las operaciones de Entrada/Salida que proporciona Matlab (`fwrite`, `fprintf`, etc.).

Recuperar datos

- Ingresar (manualmente) o pegar datos en la Ventana de Comandos.
- Crear un script para inicializar matrices u otras estructuras de datos.
- Cargar archivos binarios o ASCII utilizando `load`.
- Cargar planillas de cálculo, datos científicos, imágenes o audio con funciones que vienen con las herramientas de Matlab.
- Cargar datos desde un archivo utilizando las operaciones de Entrada/Salida que proporciona Matlab (`fread`, `fscanf`, etc.).

Guardar datos usando `save`

- Permite guardar variables del espacio de trabajo en el disco.
- Sintaxis:

```
save
save nombre_de_archivo
save nombre_de_archivo variables
save nombre_de_archivo opciones
save nombre_de_archivo variables opciones
save ('nombre_de_archivo', 'var1', 'var2', …)
```

Guardar datos usando `save`

- `save`

  Guarda todas las variables en el archivo `matlab.mat`.

- `save` nombre_de_archivo

  Guarda todas las variables en el archivo especificado.

- `save` nombre_de_archivo *variables*

  Guarda solamente las variables especificadas (separadas por espacios) en el archivo especificado.

- `save` nombre_de_archivo *opciones*

  Guarda todas las variables en el archivo especificado, utilizando alguna de las siguientes opciones:

`-append`	Agrega nuevas variables a un archivo preexistente.
`-ascii`	Guarda datos en formato ASCII con números de hasta 8 dígitos.
`-ascii -tabs`	Ídem anterior, pero las columnas se separan con tabuladores.
`-ascii -double`	Guarda datos en formato ASCII con números de hasta 16 dígitos.
`-ascii – double -tabs`	Ídem anterior, pero las columnas se separan con tabuladores.
`-mat`	Guarda datos en formato binario (el que viene por defecto)

Guardar datos usando `save`

- Ejemplo 1

```
save test.mat
```

- Ejemplo 2

```
p = 35;
q = ones(20);
save arch.mat p q
```

- Ejemplo 3

```
M = [1 2 3; 4 5 6; 7 8 9];
save arch.dat M -ascii -double
```

- Ejemplo 4

```
save('d:\pub\arch.dat','p','q','-ASCII')
```

Formatos de archivo

- Binario
  - Los datos (numéricos o alfabéticos) se guardan en representación binaria.
  - Los datos numéricos se convierten a punto flotante de máxima precisión y se pasan a binario.
  - Puede guardarse y recuperarse más de una variable.
- ASCII
  - Los datos (solo números) se guardan en representación ASCII.
  - Los números se guardan como texto. La precisión por defecto es simple, pero se pueden guardar números de doble precisión.
  - Por ejemplo, el número 27 se guarda como el texto "2.7000000e+001" (ocupa 14 bytes) en precisión simple y "2.70000000000000e+001" (ocupa 23 bytes) en precisión doble.
  - Puede guardarse una única variable.
  - Las columnas se separan por espacios o tabuladores.

Formatos de archivo

- Ejemplo de un archivo en formato ASCII

```
M = [1 2 3; 4 5 6; 7 8 9];
save arch.dat M -ascii -single -tabs
```

- Archivo `arch.dat`:

```
1.0000000e+000    2.0000000e+000    3.0000000e+000
4.0000000e+000    5.0000000e+000    6.0000000e+000
7.0000000e+000    8.0000000e+000    9.0000000e+000
```

Cargar datos usando `load`

- Permite cargar variables desde el disco.
- Sintaxis:

```
load
load nombre_de_archivo
load nombre_de_archivo variables
load -ascii nombre_de_archivo
load -mat nombre_de_archivo
load('arg1', 'arg2', 'arg3', …)
```

Cargar datos usando `load`

- `load`

  Carga las variables guardadas en el archivo `matlab.mat`.
- `load` nombre_de_archivo

  Carga las variables desde el archivo especificado.
- `load` nombre_de_archivo *variables*

  Carga solamente las variables especificadas (separadas por espacios) desde el archivo especificado.
- `load` -ascii nombre_de_archivo

  Carga las variables desde el archivo especificado, tratándolo como si su contenido estuviera en ASCII.
- `load` -mat nombre_de_archivo

  Carga las variables desde el archivo especificado, tratándolo como si su contenido estuviera en binario.

Cargar datos usando `load`

- Ejemplo 1

```
load test.mat
```

- Ejemplo 2

```
load -ascii arch.dat
```

- Ejemplo 3

```
p = load('-ASCII','d:\pub\arch.dat')
```

Otros delimitadores de columna

- Con `save` y `load` en formato ASCII las columnas se delimitan con espacios o tabuladores.
- Si quiero utilizar otros delimitadores uso `dlmwrite` y `dlmread`.
- Si quiero trabajar con archivos separados por comas (CSV files), también puedo usar `csvwrite` y `csvread`.
- Las funciones `csvwrite` y `csvread` son casos particulares de `dlmwrite` y `dlmread`.

Otros delimitadores de columna

- Ejemplos:

```
M = [1 2 3; 4 5 6; 7 8 9];
dlmwrite('arch.dat', M, '|')
```

- Archivo `arch.dat`:

```
1|2|3
4|5|6
7|8|9
```

Entrada y Salida de bajo nivel

- Matlab ofrece operaciones que permiten tener mayor control en la Entrada y Salida de datos:

`fopen`

Abre un archivo.

`fprintf`

Da formato a los datos y los escribe en un archivo.

`fscanf`

Lee datos con formato de un archivo.

`feof`

Verifica que no se haya llegado al final del archivo.

`fclose`

Cierra un archivo abierto.

Entrada y Salida de bajo nivel

- Receta para escribir un archivo:

```
1:      Abro el archivo para escribirlo.
2:      Mientras haya datos para guardar,
2.1:            Guardo los datos.
3:      Cierro el archivo.
```

- Receta para leer un archivo:

```
1:      Abro el archivo para leerlo.
2:      Mientras haya datos para leer,
2.1:            Leo los datos.
2.2:            Realizo operaciones con los datos.
3:      Cierro el archivo.
```

Entrada y Salida de bajo nivel

- Receta para escribir un archivo:

1:	Abro el archivo para escribirlo.	← `fopen`
2:	Mientras haya datos para guardar,	
2.1:	Guardo los datos.	← `fprintf`
3:	Cierro el archivo.	← `fclose`

- Receta para leer un archivo:

1:	Abro el archivo para leerlo.	← `fopen`
2:	Mientras haya datos para leer,	← `feof`
2.1:	Leo los datos.	← `fscanf`
2.2:	Realizo operaciones con los datos.	
3:	Cierro el archivo.	← `fclose`

Entrada y Salida de bajo nivel

- Consideraciones:
  - La operación `fopen` retorna un número (denominado *handler*) que representa al archivo abierto. Las demás operaciones utilizan ese *handler* para trabajar con el archivo.
  - Una vez abierto el archivo, todas las operaciones `fprintf` agregan nuevos datos.
  - Los archivos poseen un *puntero de lectura*. Cada vez que se invoca la operación `fscanf`, el puntero avanza tantos bytes como datos se hayan leído.
  - Se puede consultar si el puntero llegó al final del archivo (*end of file*, o eof) utilizando la operación `feof`.
  - Hay que recordar siempre cerrar los archivos con `fclose`.

Entrada y Salida de bajo nivel

- Sintaxis de `fopen`:

```
handler = fopen(nombre_de_archivo)
handler = fopen(nombre_de_archivo, permiso)
```

Entrada y Salida de bajo nivel

- **`handler = fopen(`**nombre_de_archivo**`)`**

  Abre el archivo para acceso de lectura.

- **`handler = fopen(`**nombre_de_archivo, tipo_de_acceso**`)`**

  Abre el archivo para el tipo de acceso especificado. Los tipos de acceso son:

`'r'`	Abrir el archivo para leerlo.
`'w'`	Abrir el archivo, o crearlo, para escribirlo. Si existe contenido, descartarlo.
`'a'`	Abrir el archivo, o crearlo, para escribirlo. Agregar nuevo contenido al final del archivo.

Entrada y Salida de bajo nivel

- Sintaxis de `fprintf`:

```
count = fprintf(handler, formato, A, ...)
```

- Descripción:

  Da formato a los datos de la matriz A (y todas las que se especifiquen) y escribe en el archivo.

  Retorna la cantidad de bytes que se escribieron.

Entrada y Salida de bajo nivel

- ¿Qué es el formato?

  El formato es un texto (*string*) que contiene caracteres ordinarios y caracteres especiales, denominados *caracteres de conversión*.

  También puede contener *caracteres de escape*.

- Caracteres de conversión

  Se utilizan para controlar la notación, alineación, dígitos significativos, ancho del campo y otros aspectos de un valor escalar o una matriz.

  Comienzan siempre con %.

- Caracteres de escape

  Se utilizan para representar teclas de control del teclado. Ejemplos de teclas de control: *enter*, *tab*, *esc*, etc.

  Comienzan siempre con \.

Entrada y Salida de bajo nivel

- Ejemplos de formatos

```
M = [1 2 5.2; -5 0 3.2; 6 -1.1 0.4]
p = 425.000641
q = 5.9990012
s = 65
```

Si se aplica el formato	Sobre los datos	Se obtiene el texto
`'%6.2f'`	p	`425.00`
`'%6.2f %6.0f'`	p, q	`425.00      6`

`'%4.0f %4.2f %4.0f\n'`	M	`1 -5.00   6`
		`2 0.00   -1`
		`5 3.20    0`

`'%c'`	s	A
`'valor de p = %6.2f'`	p	`valor de p = 425.00`

Entrada y Salida de bajo nivel

- Algunos caracteres de conversión

Carácter de conversión	Descripción
%c	Carácter
%f	Notación de punto fijo
%e	Notación exponencial

- Algunos caracteres de escape

Carácter de escape	Descripción
\b	Retroceso (borrar)
\n	Nueva línea (enter)
\t	Tabulador horizontal (tab)
\\	Barra (\)
%%	Porcentaje (%)

Entrada y Salida de bajo nivel

- Sintaxis de `fscanf`:

```
A = fscanf(handler,  formato)
A = fscanf(handler, formato, cantidad)
```

Entrada y salida de bajo nivel

- Descripción de **`fscanf`**
  - **`A = fscanf(`**handler**`,`** formato**`)`**

    Lee todos los datos (llega hasta el final) del archivo y, luego de convertirlos de acuerdo al formato especificado, los retorna en la matriz A.
  - **`A = fscanf(`**handler, formato, cantidad**`)`**

    Lee en A solamente la cantidad de datos especificados.

    Las opciones válidas para la cantidad son:

`n`	Lee como máximo n números, caracteres o strings.
`inf`	Lee hasta el final del archivo.
`[m,n]`	Lee como máximo m*n números, caracteres o strings. Llena una matriz con m filas como máximo. El valor de n puede ser inf, pero no así el valor de `m`.

Entrada y Salida de bajo nivel

- Los caracteres de conversión válidos para `fscanf` son

Carácter de conversión	Descripción
%c	Secuencia de caracteres. El número está especificado por el ancho del campo (ej.: `'%10c'` lee 10 caracteres)
%d	Enteros en base decimal
%f	Números de punto flotante
%s	Palabras (series de caracteres juntos hasta llegar al espacio en blanco)

Entrada y Salida de bajo nivel

- Sintaxis de `fclosef`:

```
estado = fclosef(handler)
```

- Descripción:

  Cierra el archivo y retorna 0 si no han ocurrido errores y -1 en caso contrario.

Entrada y Salida de bajo nivel

- Ejemplo de escritura:

```
x = 0:.1:1;
y = [x; exp(x)];
handler = fopen('datos.txt','w');
fprintf(handler, 'Valores de x, exp(x)');
fprintf(handler, '\n');
fprintf(handler,'%6.2f\t%12.8f\n',y);
fclose(handler);
```

Entrada y Salida de bajo nivel

- Archivo `datos.txt`:

```
Valores de x, exp(x)
  0.00    1.00000000
  0.10    1.10517092
  0.20    1.22140276
  0.30    1.34985881
  0.40    1.49182470
  0.50    1.64872127
  0.60    1.82211880
  0.70    2.01375271
  0.80    2.22554093
  0.90    2.45960311
  1.00    2.71828183
```

Entrada y Salida de bajo nivel

- Ejemplo de lectura:

```
handler = fopen('datos.txt','r');
s = '';
t = ' ';
while (t ~= sprintf('\n'))
  t = fscanf(handler, '%c', 1);
  s = [s t];
end
M = [];
while (~feof(handler))
    x = fscanf(handler, '%f %f', [1 2]);
    M = [M;x];
end
fclose(handler);
```

Entrada y Salida de bajo nivel

- Resultado:
  - En `t` queda guardado el último carácter que se leyó en el primer `while` (el carácter de nueva línea).
  - En `s` queda guardado el texto de la primera línea del archivo, incluyendo el carácter de nueva línea.
  - En `M` queda guardada la matriz.

Resumen

- Guardar datos

Función	Para guardar	Delimitadores	Notas
`csvwrite`	Datos numéricos	Coma	Para utilizar con planillas de cálculo (tipo Excel).
`dlmwrite`	Datos numéricos	Cualquiera	Fácil de usar, flexible.
`fprintf`	Datos numéricos y alfabéticos	Cualquiera	Parte de las rutinas de E/S. Es la más flexible, pero también la más difícil de usar.
`save`	Datos numéricos	Tabs o espacios	Fácil de usar.

Resumen

- Cargar datos

Función	Para cargar	Delimitadores	Notas
`csvread`	Datos numéricos	Coma	Para utilizar con planillas de cálculo (tipo Excel).
`dlmread`	Datos numéricos	Cualquiera	Fácil de usar, flexible.
`fscanf`	Datos numéricos y alfabéticos	Cualquiera	Parte de las rutinas de E/S. Es la más flexible, pero también la más difícil de usar.
`load`	Datos numéricos	Tabs o espacios	Fácil de usar.

Preguntas

- ¿Para qué puede servir guardar variables en un archivo en formato binario? ¿Y guardarlo en formato ASCII?
- ¿Qué ocurre si intento cargar un archivo ASCII donde se guardó más de una variable?
- ¿Cómo hago para pasar datos de una planilla de cálculo a una variable del entorno de trabajo?
- ¿Cuándo puede resultar necesario utilizar Entrada y Salida de bajo nivel?