PRÁCTICA 1 – CTE I 2020

INTRODUCCIÓN AL LABORATORIO

A. INTRODUCCIÓN A MATLAB.

1) Introducción

El programa MatLab (abreviatura de Matrix Laboratory) es una potente herramienta de cálculo numérico y visualización gráfica de uso muy difundido entre los científicos para el desarrollo de sus investigaciones. Tiene la gran ventaja de ser un lenguaje de alto nivel que integra, en un único ambiente software, rutinas de cálculo, visualización y programación. El programa es de fácil uso ya que los problemas se pueden formular usando una notación matemática standard. La representación básica de los datos en MatLab es en forma *matricial*. Algunos de los usos más comunes de MatLab son, por ejemplo:

- Cálculo numérico
- Desarrollo de algoritmos
- Modelado, simulación y desarrollo de prototipos
- Análisis y visualización de datos
- Construcción de gráficas

MatLab es un sistema abierto al cual el usuario puede incorporar nuevas funciones para su uso en aplicaciones particulares. Existen también extensiones de MatLab denominadas Toolboxes, que son librerías de funciones MatLab que permiten resolver problemas específicos en diversas áreas de ciencia e ingeniería, tales como: Control, Procesamiento de Señales, Identificación, Procesamiento de Imágenes, Redes Neuronales, Wavelets, etc.

En esta primera práctica, esperamos que el estudiante se familiarice con los comandos básicos de MatLab de forma de poder realizar el tratamiento de los datos obtenidos en las prácticas siguientes.

IMPORTANTE: Dado el carácter a distancia del curso en el presente año, y considerando que MatLab es software propietario, se utilizará OCTAVE, una implementación de software libre de las principales funciones y paquetes de MatLab.

Al iniciar el programa OCTAVE se desplegará una ventana desde donde se ejecutan los diferentes comandos (ver Figura 1.1).

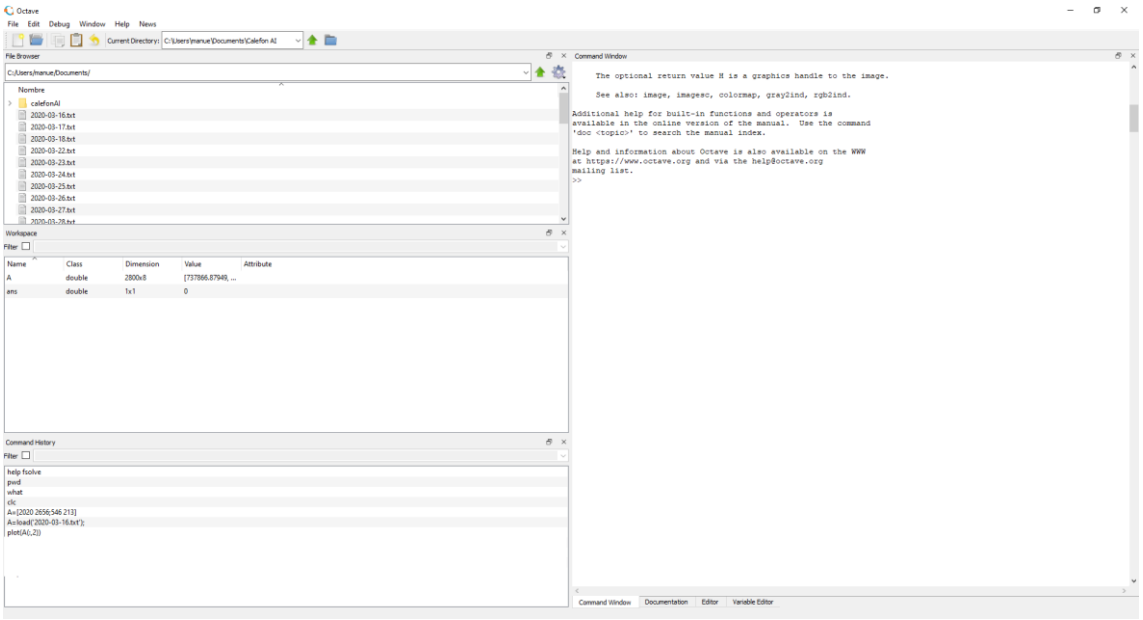


Figura 1.1*:* Escritorio de trabajo típico de OCTAVE.

2) Comandos Básicos

El símbolo >> (llamado terminal, consola, prompt o línea de comandos) se muestra en la Ventana de Comandos (Command Window) y es el lugar donde el usuario ingresa a MatLab/OCTAVE los comandos para que sean ejecutados. Antes de comenzar a trabajar es conveniente cambiarse al directorio de trabajo (ver Figura 2.1). Siempre que se quiera acceder a más información acerca de las tareas de MatLab y toolboxes, podemos ir al menú Help / MatLab Help.

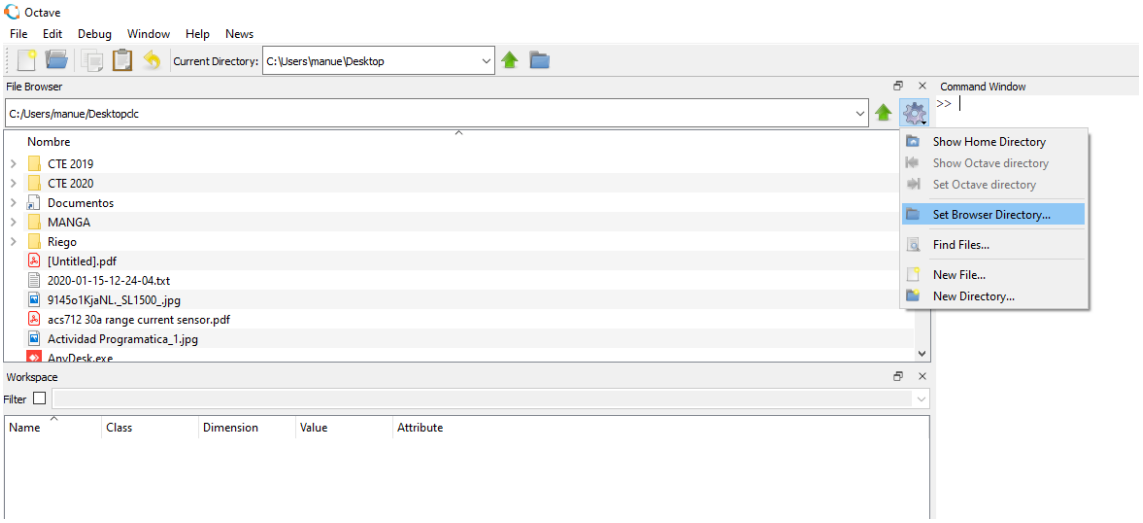


Figura 2.1: Forma de cambiar de directorio de trabajo del usuario (Browser Directory).

En el cuadro 2.1, veremos algunos comandos básicos de MatLab/OCTAVE.

Comando	Descripción
help	Lista la serie de funciones o aplicaciones que abarca Matlab (*)
pwd	Indicia el directorio en el cual se está trabajando
demo	Permite acceder a las demostraciones de algunas funciones de Matlab (*)
what	Lista los M-Files existentes en el disco (*)
whos	Lista las variables que se encuentran en la memoria y sus características (*)
clear	Limpia todas las variables de la memoria
clc	Limpia el texto de la Ventana de Comandos

Cuadro 2.1: Comandos básicos de ayuda y navegación en MatLab. Los comandos marcados con (*) tiene sentido utilizarlos solamente en la Ventana de Comandos, no en los M-Files.

3) Vectores y Matrices

Veamos ahora cómo construir vectores y matrices en MatLab/OCTAVE. Esto es fundamental pues los datos deben ser presentados en forma de matrices para que el programa pueda procesarlos. En la Figura 3.1 (superior-izquierda e inferior-izquierda) se muestra cómo se definen una matriz (A) y un vector (v). El comando *size* indica las dimensiones de la matriz: el primer número corresponde a la cantidad de filas y el segundo a la cantidad de columnas (ver Figura 3.1 superior-derecha). Si queremos identificar un elemento determinado de una matriz solo tenemos que indicar su fila y columna, como se muestra en la Figura 3.1 (superior-derecha).

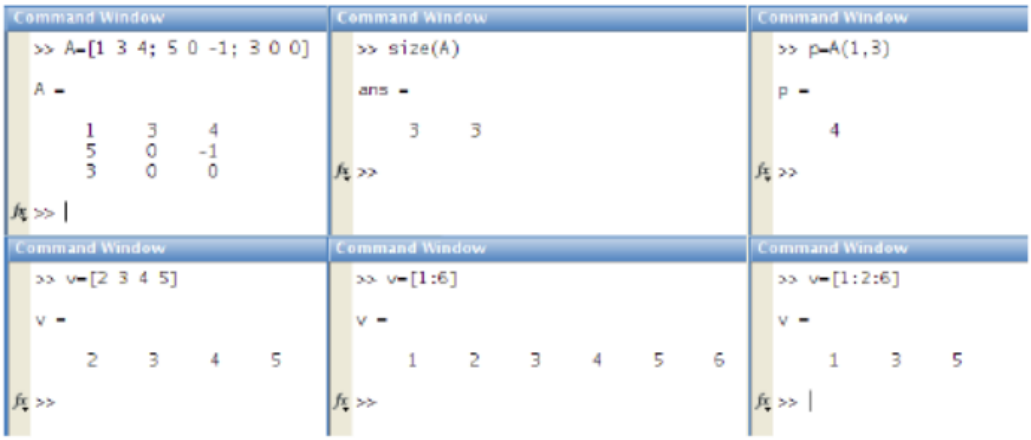


Figura 3.1: *Superior-Izquierda:* En el ejemplo se define una matriz A de 3 filas y 3 columnas (3 x 3), escribiendo los elementos que constituyen sus filas. *Superior-Centro:* El comando *size* indica las dimensiones de la matriz A. Para vectores el comando adecuado es *length*. *Superior- Derecha*: Se muestra la forma de llamar un elemento dado de la matriz A. *Inferior-Izquierda*: Se muestra la forma en que se definen los vectores. *Inferior-Centro*: Se muestra la forma de definir un vector indicando el comienzo y el final. *Inferior-Derecha*: Un comando similar permite generar un vector indicando el comienzo, final y el paso de incremento entre sus elementos, de esta forma se genera un vector de 1 a 6 con incremento 2.

3.1. Formas de definir Vectores

Los vectores se definen como cualquier *n-upla* de elementos en MatLab/OCTAVE. La cantidad de elementos determina su dimensión y se pueden definir explícitamente de tres

formas: como se muestra en la Figura 3.1 (inferior-izquierda), indicando los elementos inicial y final (Figura 3.1 inferior-centro), o de forma incremental (Figura 3.1 inferior derecha). Los vectores también pueden ser definidos a partir de las filas o columnas de una matriz: simplemente se indica la fila o columna a llamar, utilizando el símbolo : (dos puntos) para indicar el índice recorrido y un número para indicar el índice fijo (ver Figura 3.2 izquierda).

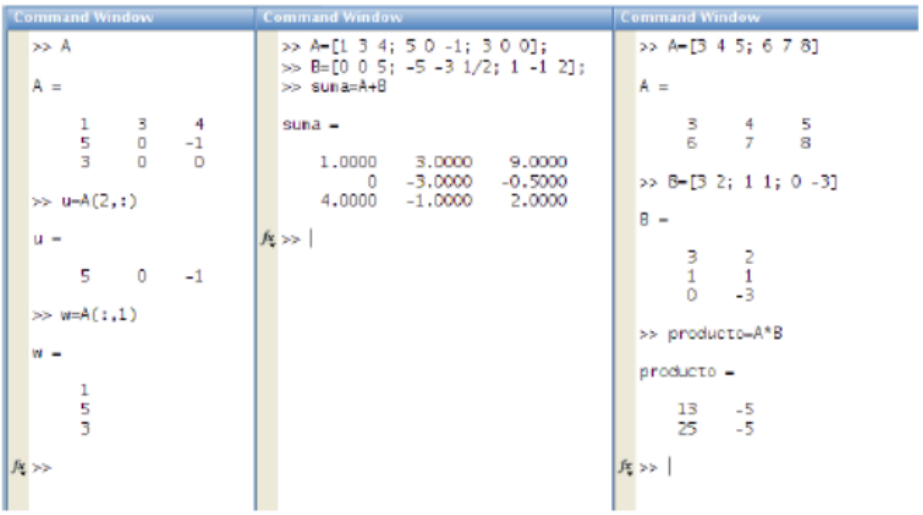


Figura 3.2*: Izquierda:* Diferentes formas de definir vectores a partir de una matriz cualquiera. *Centro:* Operación de suma de matrices, observe que dichas matrices deben ser de iguales dimensiones. *Derecha:* Operación de producto de matrices. Observe que el número de columnas de A es igual al número de filas de B.

3.2. Suma de matrices

Tanto la suma como la resta solamente puede aplicarse a dos o más matrices de iguales dimensiones y la misma se realiza elemento a elemento como lo muestra la Figura 3.2 (centro). El resultado de la suma de dos matrices es *otra matriz* (de igual dimensión que las matrices sumando).

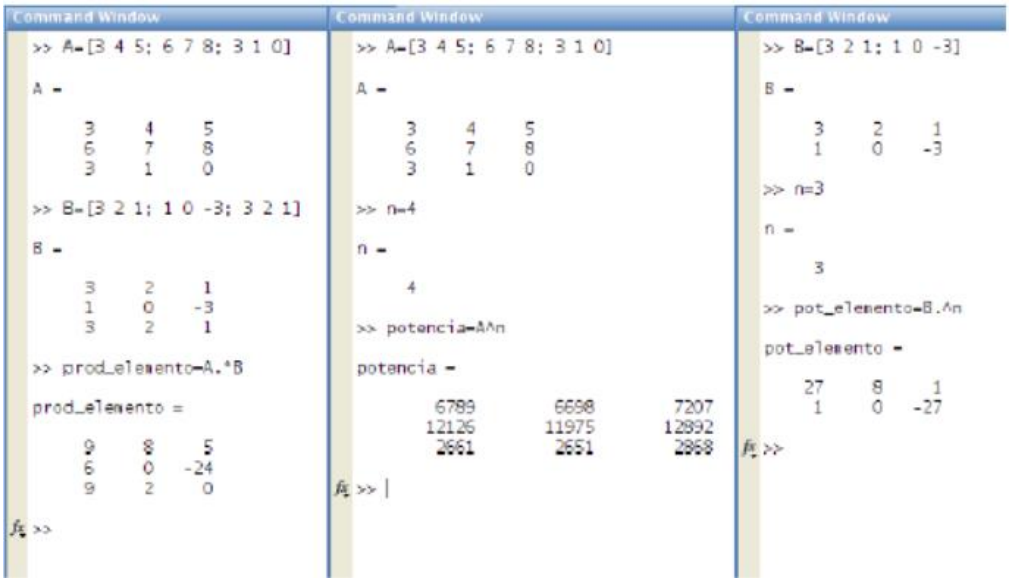


Figura 3.3*: Izquierda*: Operación de producto de matrices elemento a elemento*. Centro*: Potencia de una matriz. Observe que equivale al producto usual de la matriz con ella misma, por lo tanto tiene que ser cuadrada. *Derecha:* Potenciación elemento a elemento de una matriz.

3.3. Producto de Matrices

Para efectuar el producto de las matrices A y B se debe cumplir que el número de columnas de A sea igual al número de filas de B como se muestra en la Figura 3.2 (derecha). Es claro que tal condición se satisface en matrices cuadradas. En ese caso particular también es posible definir otro producto conocido como Producto Elemento a Elemento. El mismo da un resultado diferente al producto usual de matrices y tiene aplicaciones varias en la manipulación de datos. Como el nombre lo dice, el producto es elemento a elemento de forma que cada elemento está definido por el producto de los elementos correspondientes en las matrices factor. Observe que entre el símbolo de producto * (asterisco), se antepone un . (punto) para diferenciarlo del producto usual de matrices. El resultado de cualquier producto entre matrices es *otra matriz*, de dimensión n x m, donde n es el numero de filas de A y m es el numero de columnas de B.

Producto de Vectores

Para efectuar el producto (*escalar*) de dos vectores v y s (*v*s*), se debe cumplir que ambos tengan la misma dimensión. Además, para ejecutarlo en MatLab/OCTAVE es necesario que el primero sea 'vector fila' y el segundo 'vector columna'; siempre dependiendo de cómo se hayan definido, esto se puede lograr *transponiendo* a *v* o *s*. El resultado del producto entre vectores puede ser tanto un *número* como una *matriz* o un *vector*; para la operación que consideramos aquí (vector fila x vector columna) el resultado es un *número*; para el producto elemento a elemento o el *producto vectorial*, el resultado es otro *vector*.

Todo lo anterior referido al producto entre matrices y vectores es válido igualmente para la división entre matrices (siempre que el determinante de la matriz divisor sea no nulo), y la división entre vectores (por ser el vector un caso particular de matriz).

3.4. Potencia Enésima de una Matriz

Algo similar a lo expuesto para el producto de matrices. La potenciación usual se denota con el símbolo ^ (“techo”) como se muestra en la Figura 3.3 (centro). La matriz a elevar tiene que ser cuadrada. Análogamente la potenciación elemento a elemento simplemente opera elevando a la potencia cada elemento por separado y es aplicable a cualquier tipo de matrices. Nuevamente para indicar que la operación es elemento a elemento se antepone un . (punto) antes del ^ (“techo”), ver Figura 3.3 (derecha).

4) Lectura y Almacenamiento de Datos

4.1. Almacenamiento de Datos

MatLab/OCTAVE permite varias opciones para almacenar las variables con las cuales se trabaja para su posterior utilización. En todos los casos el comando es *save* pero la sintaxis varía de acuerdo a la forma que queramos guardarlo, ya sea en formato ASCII (ver Figura 3.4 superior-izquierda), o formato binario de MatLab/OCTAVE (ver Figura 3.4 inferior-izquierda).

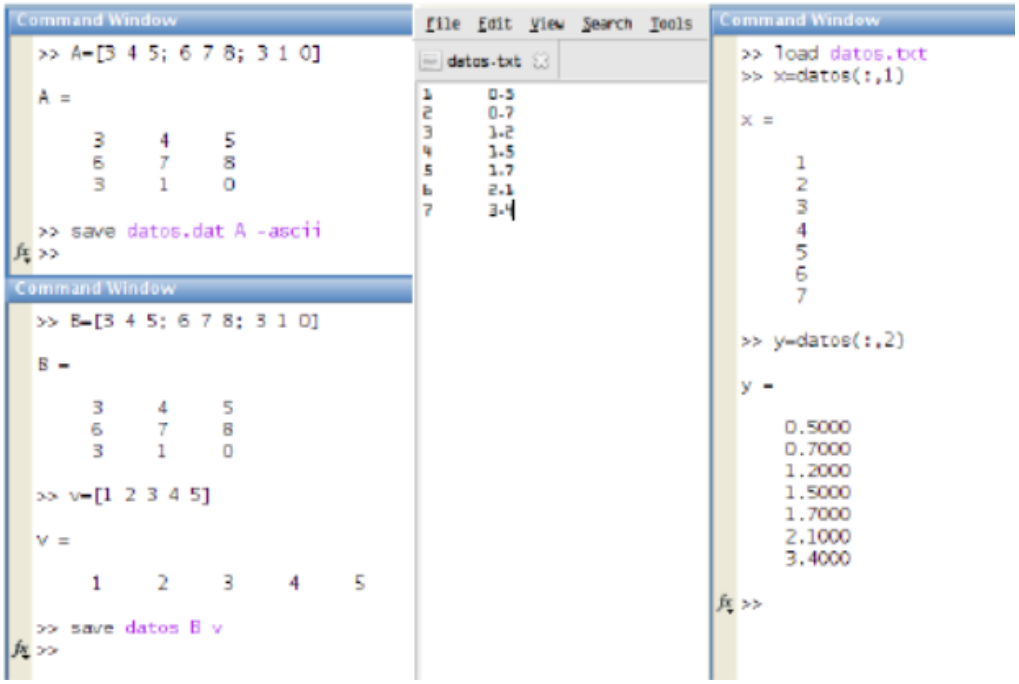


Figura 3.4: *Superior-Izquierda*: Sintaxis para guardar una sola variable a la vez en formato de texto ASCII. *Inferior-Izquierda*: Sintaxis para guardar una o más variables en formato binario de MatLab/OCTAVE. *Centro*: Vista de un procesador de texto plano (como el Block de Notas) donde puede generarse un archivo de datos en formato ASCII. *Derecha:* Comando para cargar los datos en MatLab/OCTAVE y asignar como vectores las distintas columnas de la matriz de datos.

4.1.1. Almacenamiento en formato de texto ASCII

La sintaxis es

save <nombre del archivo> <variables> -ascii

La ventaja de usar este método radica en que este archivo de datos puede ser leído por cualquier programa de manejo de texto y/o planillas de cálculo, por ejemplo: Block de Notas, Excel, etc. La terminación .dat no es obligatoria (también se suele usar la terminación .txt), pero se suele utilizar .dat para identificar rápidamente el archivo como un archivo de datos en ASCII. El mayor inconveniente que tiene este método es que todas las variables deben tener la misma dimensión para ser almacenadas.

4.1.2. Almacenamiento en formato binario de MatLab/OCTAVE

La sintaxis es

save <nombre del archivo> <variables>

Por defecto, MatLab/OCTAVE coloca a estos archivos la terminación .mat para indicar que el formato es de MatLab/OCTAVE. Estos archivos no pueden ser leídos desde programas de procesamiento de texto, pero tienen algunas ventajas que se verán más adelante. En este caso no es necesario que todas las variables tengan la misma dimensión para ser almacenadas.

4.2. Creación manual de un archivo de datos

Suele ocurrir que los datos experimentales que se obtienen en el laboratorio son anotados en libretas o cuadernos que suelen ser pasados al PC. Una forma cómoda para poder utilizar estos datos posteriormente con MatLab/OCTAVE es generar un archivo de datos desde un editor de texto elemental o texto plano (o en inglés "plain text") como el Block de Notas (ver Figura 3.4 centro). Los archivos generados como texto plano son de formato ASCII. Para generarlo simplemente se debe abrir el Block de Notas e ingresar los datos en forma de columnas separadas por espacios o tabulaciones. Al almacenar tener la precaución de ponerle terminación .dat o .txt para identificarlos rápidamente como archivo de datos.

4.3. Lectura de Archivos de Datos

El comando para leer archivos de datos es *load* y también difiere en la sintaxis según se trate de un archivo de datos en formato ASCII (Figura 3.4 derecha) o binario de MatLab/OCTAVE.

4.3.1. Archivos ASCII

La sintaxis es

```
load <nombre del archivo>
```

Notemos que si hacemos un *whos* la variable que tenemos en la memoria del MatLab/OCTAVE tiene el mismo nombre que el archivo, pero sin la terminación.

4.3.2. Archivos binarios de MatLab/OCTAVE

La sintaxis es

```
load <nombre del archivo>
```

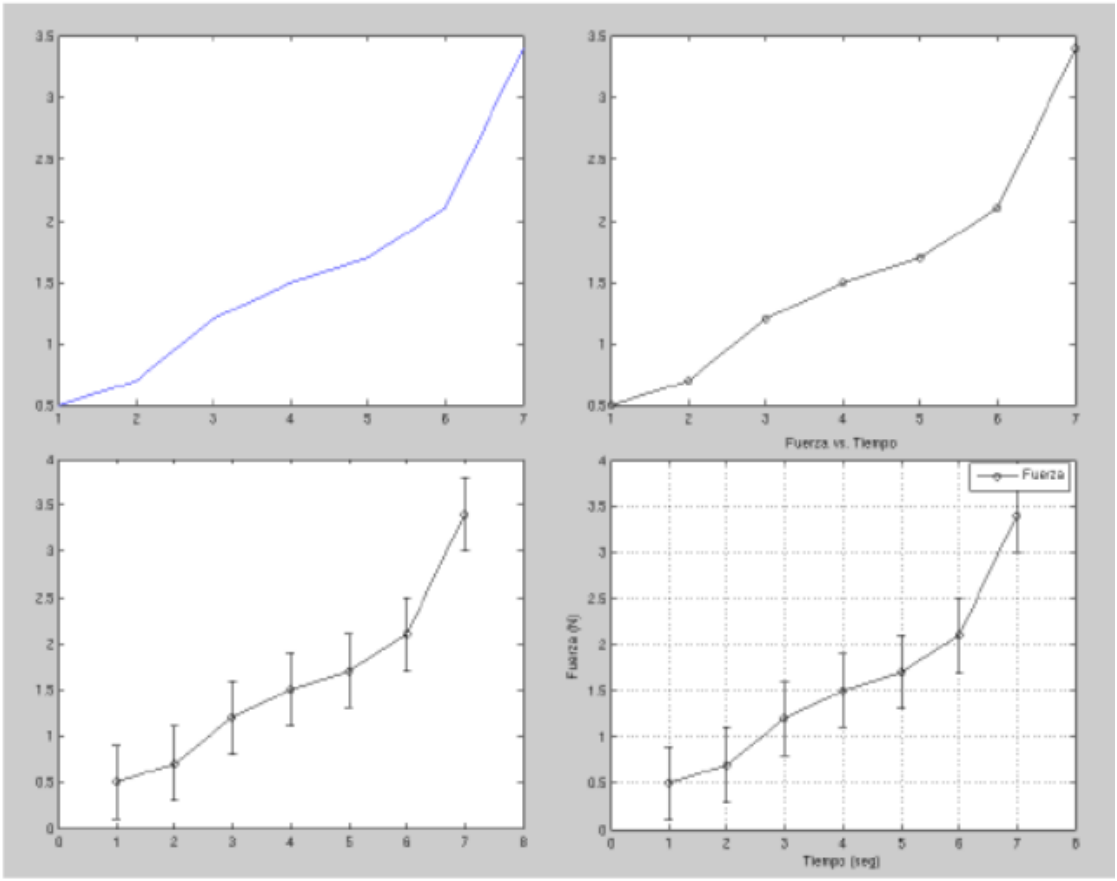
Otra opción para las versiones más nuevas es

a = load('nombre del archivo')

De esta forma la variable contiene los datos almacenados en el archivo. Notemos que si hacemos un *whos*, las variables que tenemos en la memoria del MatLab/OCTAVE tiene el mismo nombre con que habían sido almacenadas, lo que representa una gran ventaja si se está realizando la lectura en el marco de un programa, puesto que se conoce a priori el nombre de las variables.

5) Gráficas

MatLab/OCTAVE tiene un excelente manejo de gráficos. Aquí veremos sólo algunos de los comandos básicos para la generación de gráficas en dos dimensiones.



PLOTEO	COMANDOS INGRESADOS
SUPERIOR-IZQUIERDA	`plot(x,y)`
SUPERIOR-DERECHA	`plot(x,y,'ok-')`
INFERIOR-IZQUIERDA	`errorbar(x,y,e,'ok-')`

INFERIOR-DERECHA	`errorbar(x,y,e,'ok-')`
	`title('Fuerza vs. Tiempo')`
	`xlabel('tiempo (seg)')`
	`ylabel('Fuerza (N)')`
	`legend('Fuerza')`
	`grid on`

Figura 5.1: *Superior-Izquierda*: Ploteo simple de dos vectores. *Superior-Derecha*: Ploteo con opción de color, punto y línea. *Inferior-Izquierda*: Ploteo con opciones de color, punto, línea, y barra de error. *Inferior-Derecha*: Ploteo con opciones de color, punto, línea, barra de error, grilla, titulo, leyenda y nombre en los ejes X e Y. *Inferior*: Tabla donde se especifican los comandos y sintaxis ingresados para obtener los distintos ploteos.

5.1. Comandos para Elaborar Gráficos

En el Cuadro 5.2 se listan una serie de comandos gráficos. Algunos pueden ser utilizados en secuencia para graficar una serie de datos y obtener una gráfica final con todas las leyendas y etiquetas correspondientes, como se muestra en la tabla de comandos de la Figura 5.1. Observe que para ejecutar el comando *errorbar*, es necesario un tercer vector *e*, de igual longitud que *x* e *y*, que contiene el error de *y*.

COMANDO	**DESCRIPCION**
figure	Genera una nueva pantalla de gráficos
plot	Crea un gráfico simple
plotyy	Permite plotear dos sets de datos, ambos con ejes de las Y a la izquierda y derecha
loglog	Permite graficar en escala logarítmica en ambos ejes
semilogx	Permite graficar en escala logarítmica en el eje de las X y lineal en el eje de las Y
semilogy	Permite graficar en escala logarítmica en el eje de las Y y lineal en el eje de las X
subplot	Permite dividir la ventana grafica en cuadrantes y hacer varias graficas en el
ezplot	Permite crear un gráfico de variable simbólica
errorbar	Crea un gráfico con barras de error
hold	Permite realizar la superposición de dos o más gráficos en una misma pantalla
grid	Activa o desactiva la grilla automática en la gráfica
title	Ingresa un título en la gráfica
xlabel	Ingresa la etiqueta en el eje de las X
ylabel	Ingresa la etiqueta en el eje de las Y
text	Permite ingresar texto en la gráfica en coordenadas definidas por el usuario
axis	Permite ajustar el rango de valores en X e Y y también su proporcionalidad
legend	Ingresa una leyenda para los datos experimentales

Cuadro 5. 2: Lista de comandos gráficos en MatLab/OCTAVE. Para aprender cómo es la sintaxis del comando se debe ingresar *help* seguido del comando en la Ventana de Comandos.

6) Creación de M-Files

MatLab/OCTAVE permite ejecutar secuencias de comandos almacenados en un archivo. Estos archivos deben tener la extensión .m y por eso se denominan M-files. Existen básicamente dos tipos de M-files: los denominados *function-files* y *script-files*. La forma de editar M-files es usando un editor incorporado a MatLab/OCTAVE, el denominado MatLab Editor/Debugger, al cual se accede desde el menú Archivo.

6.1. Script files

Un script-file consiste de una sucesión de líneas de comando tal como las ingresaríamos en la Ventana de Comandos de MatLab/OCTAVE para su ejecución en tiempo real. Por ejemplo, si el archivo tiene el nombre *temp_conv.m*, como se muestra en la Figura 6.2 (superior), el mismo puede ser ejecutado desde la Ventana de Comandos y todas las líneas del programa se ejecutarán en el orden en que aparecen. Las variables definidas en el script-file son globales por lo tanto cambiarán el valor de aquellas variables que estén definidas con el mismo nombre antes de ser ejecutado el programa.

Para evitar confusiones se suele encabezar el programa con las tres líneas que aparecen en la Figura 6.2 (superior). Las mismas tienen como tarea: limpiar la pantalla, cerrar todas las ventanas gráficas y limpiar las variables de la memoria, respectivamente. Los script-files también pueden ser creados en editores de texto convencionales de los que se mencionaron anteriormente en esta práctica. Para poder ver qué es lo que el programa está haciendo en medio de la ejecución, los script-files pueden imprimir texto en la ventana de comandos con el comando *disp*, como ser mensajes de bienvenida, aviso de algún error, etc.

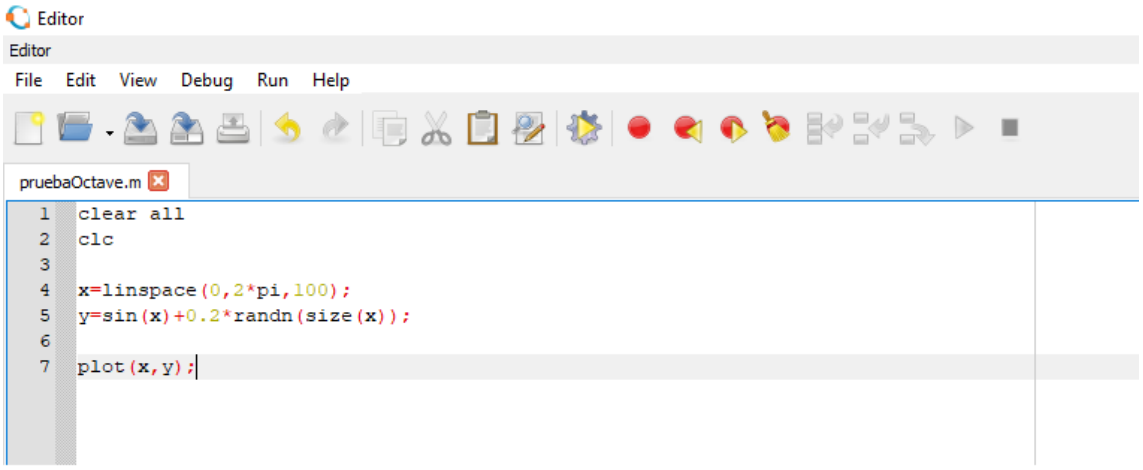
Cuando el programa es muy extenso es muy usual hacer anotaciones dentro del mismo para guiarnos durante la programación. Tales comentarios o anotaciones no deben ser interpretados por MatLab/OCTAVE a la hora de ser ejecutado el programa. Esto se logra anteponiendo el símbolo *%* (por ciento) delante del comentario que automáticamente se pintará de color verde.

En el Cuadro 6.1 se muestran una serie de comandos típicos de programación de script-files a los cuales se le suman todos los comandos gráficos, los comandos para guardar y cargar datos en archivos .dat o .txt anteriormente citados.

COMANDO	DESCRIPCION
clc	Limpia el texto de la Ventana de Comandos
clear all	Limpia todas las variables de la memoria

close all	Cierra todas las ventanas gráficas abiertas
disp	Escribe texto en la Ventana de Comandos durante la ejecución
input	Permite que el usuario ingrese un valor mediante teclado, que es almacenado como una variable
if	Abre la sentencia condicional
for	Abre la tarea de bucle o repetición
switch	Similar a IF, permite ramificar la tarea según el valor de alguna variable
end	Finaliza sentencias IF, FOR, SWITCH
menu	Permite elegir al usuario qué opción seguir ante una ramificación
export	Permite exportar datos o variables en distintos formatos ASCII
zeros	Crea una matriz de elementos todos 0
ones	Crea una matriz de elementos todos 1
eye	Crea una matriz identidad
min	Devuelve el mínimo elemento de un vector
max	Devuelve el máximo elemento de un vector
mean	Calcula el promedio de los elementos de un vector
std	Calcula la desviación estándar entorno al promedio de los elementos de un vector
mode	Calcula la moda entre los elementos de un vector
sum	Calcula la suma de todos los elementos de un vector
length	Devuelva la longitud de un vector
size	Devuelve las dimensiones de una matriz en filas y columnas
transpose	Devuelve la transpuesta de una matriz o un vector
polyfit	Realiza un ajuste por un polinomio de grado n y calcula los coeficientes del promedio
corrcoef	Calcula el coeficiente de correlación de Pearson para una serie de datos experimentales
inv	Calcula la inversa de una matriz cuadrada con determinante no nulo

Cuadro 6.1: Comandos de programación y cálculo y su descripción. Para aprender cómo es la sintaxis del comando se debe ingresar *help* seguido del comando en la Ventana de Comandos.



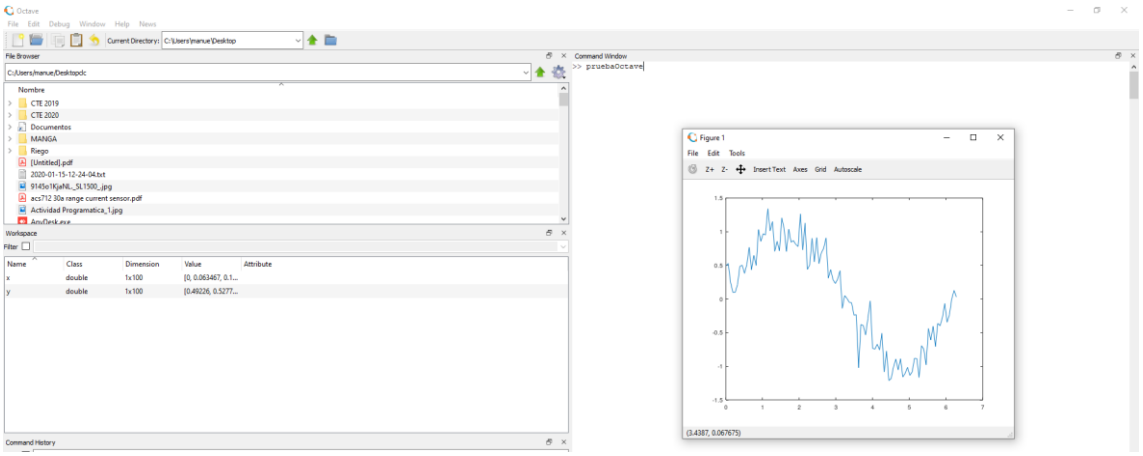


Figura 6.2: *Superior:* Vista del editor MatLab Editor/Debugger donde se muestra un programa M-File de tipo Script-File. El programa/script ha sido guardado bajo el nombre de *pruebaOctave.m*. *Inferior*: La ejecución del mismo programa en la ventana de comandos. El programa ha sido ejecutado simplemente escribiendo *pruebaOctave* en la Ventana de Comandos

7) Ejercicios

Ejercicio 1: Vectores

1. Crear un vector *v* cuyo primer elemento sea 55, el último 480 y tal que la diferencia entre dos elementos consecutivos sea 5. Obtener el número de elementos de *v* mediante un comando y definir una variable *n* que contenga tal número.
2. Definir un vector *u* que contenga la raíz cúbica de los elementos de *v*.
3. Transponer los vectores *v* y *u*.
4. Definir los siguientes vectores:
   - *q = n * v*
   - *s = v * u*
   - *t = u / v*

   Definir las variables *sumq, sums* y *sumt*, que contengan la suma de los elementos de los vectores q, s y t, respectivamente.

5. Almacenar todas las variables y vectores definidos en un archivo binario cuyo nombre sea ‘ejercicio1p1’

Ejercicio 2: Matrices

$$A = \begin{pmatrix} 3 & 0 & -2 \\ 1 & 4 & 5 \\ -1 & 1 & 2 \end{pmatrix}, B = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 6 & 1 \\ 3 & -2 & -5 \end{pmatrix}, C = \begin{pmatrix} -1 & -1 & 2 \\ 5 & 1 & 1 \\ -3 & -2 & 3 \end{pmatrix}$$

1. Dadas las matrices A, B y C, realizar las siguientes operaciones:
   I. A + B - C
   II. A * B
   III. C^2
2. Elevar cada uno de los elementos de la matriz C al cuadrado. (Notar que la matriz resultante aquí es diferente a la obtenida en el punto 1.III.)
3. Calcular la matriz inversa de A
4. Calcular el determinante de B
5. Definir una nueva matriz D tal que $d_{ij} = a_{ij} \times b_{ij}$. (Notar que la matriz D es diferente a la obtenida en el punto 1.II.)
6. Almacenar <u>todas</u> las variables en un archivo ASCII cuyo nombre sea 'ejercicio2p1'

Ejercicio 3: Procesamiento de datos

1. Lea el archivo matriz.dat
2. Determine las dimensiones de la matriz y cree una variable *n* para el número de filas y una variable *m* para el número de columnas.
3. Seleccione los elementos de la segunda fila, de la cuarta, y los de la segunda columna, y guárdelos en tres vectores.
4. Defina una matriz de dimensión *n* x *m* de ceros y una matriz de igual dimensión pero cuyos elementos sean todos 1.
5. Almacenar las variables en un archivo cuyo nombre sea 'ejercicio3p1'
6. Grafique la segunda columna en función de la cuarta y la primera columna en función de la tercera, utilizando diferentes tipo de símbolos y colores. Etiquete los ejes y asigne un título y leyenda a la gráfica.
7. Almacenar la gráfica como archivo .jpg con el nombre 'grafica3p1'

Ejercicio 4: Scripts

Desarrolle un script que resuelva ecuaciones de segundo grado ($ax^2 + bx + c = 0$) permitiendo introducir los coeficientes a, b y c mediante teclado.