2.- Introducción a Matlab

DR. SERVANDO LÓPEZ AGUAYO

AGOSTO-DICIEMBRE 2020

En este primer episodio…

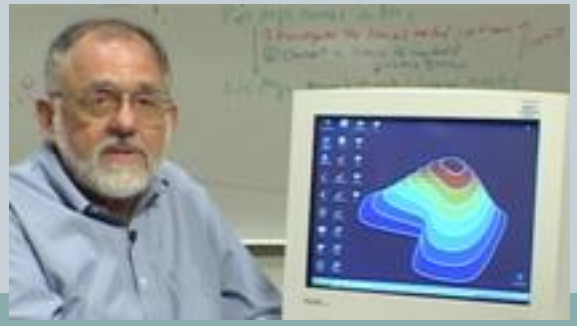
- Historia y funcionalidad de Matlab.
- Conocimiento general de la interfaz de Matlab.
- Números, variables, vectores y matrices.
- Operadores.
- Archivos .m
- Funciones básicas de programación.

¿Qué es Matlab?

- Es un lenguaje de alto rendimiento para computación especializada, con usos típicos en procesos de ingeniería y ciencias.
- Es un ambiente de programación y visualización de datos numéricos.
- Es un entorno de desarrollo de aplicaciones.
- https://www.mathworks.com/videos/technical-computing-with-matlab-69042.html

Historia de Matlab

- MATLAB = MATrix LABoratory.
- Primera Versión es en 1984.
- Desarrollado por Cleve Barry Moler.
- Tiene sus orígenes en Fortran.



Desventajas de usar Matlab

- La curva de aprendizaje!
- No es gratis.
- No es tan veloz como C o Fortran.
- Se necesita a veces de un “Toolbox” que… tristemente… cuesta extra!

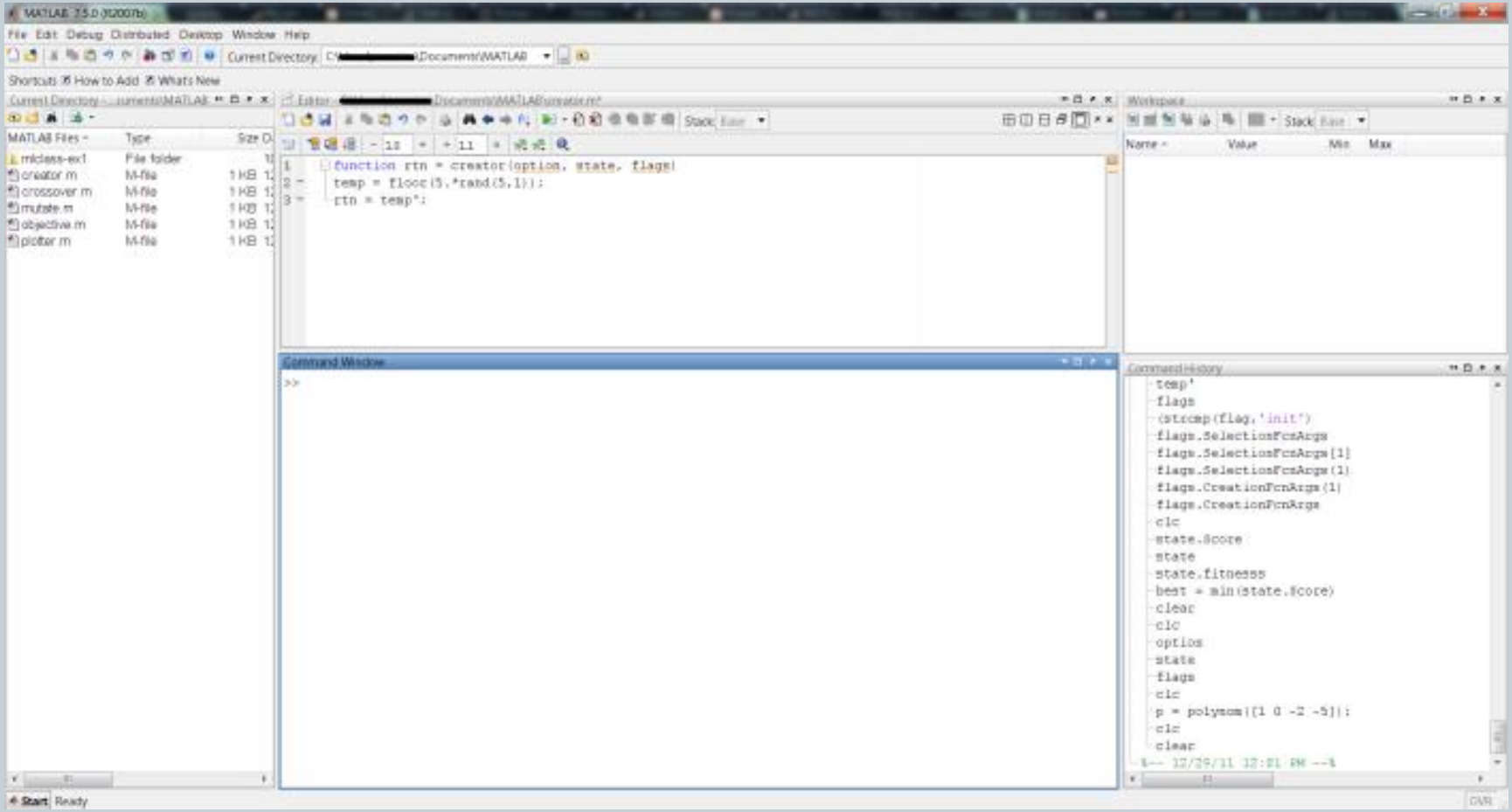
Ventajas de usar Matlab

- Más de un millón de usuarios.
- Reconocido en la academia y en la industria.
- Códigos por toda la web.
- Múltiples funciones ya elaboradas.
- Constante mejoramiento.

Tener muy presente:

- La curva de aprendizaje!

Interfaz básica de Matlab



Números en Matlab

- Números enteros: 1 25 -18
- Números reales: 1. -2.32 28.15
- Números complejos: 4+7i, 0.3*j
- Notación exponencial: 4e-7, 57.9e7
- Número de Euler: exp(1)
- Número Pi: pi

- Visualizar mayor cantidad de dígitos:

  format long / format short

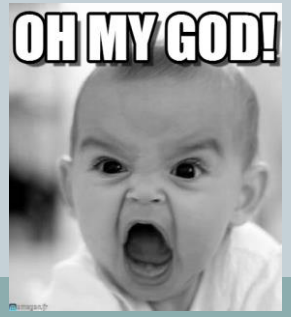
Variables en Matlab

- No es necesario “declarar variables” (por default: son matrices de doble precisión).
- Ciertas restricciones: los nombres no pueden empezar con números, ciertos operadores, etc.
- Se distingue entre mayúsculas y minúsculas.
- Se pueden reasignar valores cuantas veces sea necesario.

Algunas variables predefinidas

- pi = 3.1416
- 1i,1j = Raíz cuadrada de menos uno.
- inf = Infinito.
- nan = not a number.
- ans = resultado más reciente.

* Cuidado! Estás variables se puede redefinir momentáneamente por el usuario!
(y dar un auténtico dolor de cabeza)



Algunos comandos de Matlab

- El comando más importante: help

- Limpiar una variable: clear variable
- No desplegar resultados: ;
- Comando anterior: ↑
- Búsqueda de comandos: comando + TAB
- Limpiar pantalla: clc
- Salir de Matlab: exit

Parte central de Matlab

- Las Matrices: Usar coma o espacio para separar los elementos, y usar punto y coma para separar los reglones.

Vector renglón

```
>> A = [2,3,5]
A =  2     3     5

>> D = [2 3 5]
D =  2     3     5
```

Vector columna

```
>> B = [7; -4; 1]
B =
     7
    -4
     1
```

Matriz:

```
>> C = [1,2,3; 4,5,6]
C =
     1     2     3
     4     5     6
```

Aritmética de Matlab

- Toma en cuenta por default, el uso de números complejos y matrices, facilitando así –generalmente- nuestros códigos.

Actividad 1

a) Realiza la siguiente multiplicación de matrices

$$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix} \times \begin{bmatrix} 1 & 8 & 9 \\ 2 & 7 & 10 \\ 3 & 6 & 11 \\ 4 & 5 & 12 \end{bmatrix} =$$

b) Multiplica elemento por elemento los siguientes vectores:

$$\begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix} * \begin{bmatrix} 5 \\ 1 \\ 3 \end{bmatrix} =$$

Actividad 1

c) Reescribe los siguientes números, mostrando su parte real y parte imaginaria: exp(i*pi/4), sqrt(-4) y log(i). Hazlo de manera analítica y comprueba tu resultado con Matlab.

d) Investiga los comandos en Matlab para extraer la parte real, la parte imaginaria, dar el valor absoluto y el ángulo de los números complejos.

- Reporta tus resultados obtenidos.

Operadores en Matlab

Operadores aritméticos usuales

Suma	+
Resta	-
Multiplicación	*
División	/
Potenciación	^n
Raiz cuadrada	sqrt()

Jerarquía:

potenciación,

divisiones y multiplicaciones

sumas y restas.

Operadores relacionales usuales

Menor que	<
Menor o igual que	<=
Mayor que	>
Mayor o igual que	>=
Igual	==
Diferente	~=

Operadores lógicos

and	&
or	|
not	~

El operador doble punto

- Muy útil para la creación de vectores.
- Crear un vector del 1 al 1000 y asignarlo a la variable a:

```
a=1:1000
```

- Crear un vector del 0 al 1000 en pasos de 10 y asignarlo a la variable hola:

```
hola=0:10:1000
```

- Crear un vector del 10 al 0 en pasos de 0.5 y asignarlo a la variable vamos:

```
vamos=10:-.5:0
```

Manejo de vectores y matrices

- Crear el vector v con elementos:

  v = [16 5 9 4 2 11 7 14];

- Revisar los siguientes comandos:

  v(3); v([1 5 6]) ; v(3:7) ;

  v2 = v([5:8 1:4]) ; v(end) ;

  v(5:end); v(1:2:end);

  v([2 3 4]) = [10 15 20];v([2 3]) = 30

Manejo de matrices

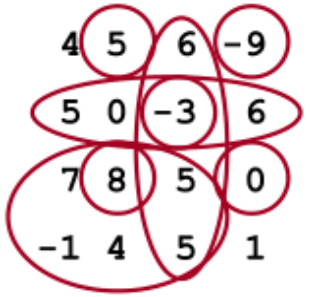
- Para el caso de una matriz de mxn

Formato por coordenadas:

```
A(renglones,columnas)
```

Sea

```
A =
```




```
>> A(2,3)
ans =
-3
```

```
>> A(2,:)
ans =
5 0 -3 6
```

```
>> A(:,3)
ans =
6
-3
5
5
```

```
>> A(3:4, 1:3)
ans =
 7 8 5
-1 4 5
```

```
>> A([1,3], [2,4])
ans =
5 -9
8  0
```

Manejo de matrices

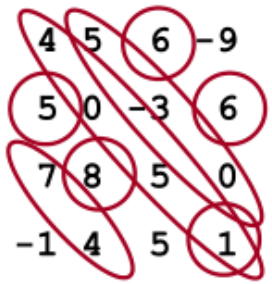
Formato unidimensional:

```
A(vector 1D de posiciones)
```

```
>> A([2,7,9,14,16])
ans =  5   8   6   6   1
```

Sea

```
A =
 4   5   6  -9
 5   0  -3   6
 7   8   5   0
-1   4   5   1
```



Diagonales

```
>> diag(A)
ans =
 4
 0
 5
 1
```

```
>> diag(A,1)
ans =
 5
-3
 0
```

```
>> diag(A,-2)
ans =
 7
 4
```

Modificación de elementos

```
A(renglones,columnas) = Nuevos_valores
A(vector 1D de posiciones) = Nuevos_valores
```

Sea

```
A =
 4 5  6 -9
 5 0 -3  6
 7 8  5  0
-1 4  5  1
```

```
>> A(2,3) = 10;
>> A(3,:) = [1,2,3,4];
>> A(:,1) = [3,7,8,0]';
>> A([1,4],[2,3]) = [8,5;2,1];
>> A([3,6,9,11,12]) = [1,6,3,9,7]
```

Agregar elementos

Sea

```
A =
   4  5   6  -9
   5  0  -3   6
   7  8   5   0
  -1  4   5   1
```

```
>> A(5,:) = [0,1,0,-1]
A =
   4  5   6  -9
   5  0  -3   6
   7  8   5   0
  -1  4   5   1
   0  1   0  -1
```

```
>> A([2,4],6) = [3;-2]
A =
   4  5   6  -9   0   0
   5  0  -3   6   0   3
   7  8   5   0   0   0
  -1  4   5   1   0  -2
```

Eliminación de elementos

Sea

```
A =
   4  5   6  -9
   5  0  -3   6
   7  8   5   0
  -1  4   5   1
```

```
>> A(:,3) = []
A =
   4  5  -9
   5  0   6
   7  8   0
  -1  4   1
```

```
>> A([2,3],:) = []
A =
   4  5   6  -9
  -1  4   5   1
```

Y ahora…

- Nuestro break! ☺ Regresamos en 10 mins!

Operadores

- Matlab tiene múltiples operaciones predefinidas:

Longitud de un vector	`length(u)`
Norma de un vector	`norm(u)`
Suma de elementos de un vector	`sum(u)`
Mult. de elementos de un vector	`prod(u)`
Tamaño de una matriz	`[m,n] = size(A)`
Transpuesta	`transpose(A)`
Transpuesta conjugada	`A'`
Determinante	`det(A)`
Inversa	`inv(A)`
Eigenvalores y eigenvectores	`[V,E] = eig(A)`

Operadores

- Realmente muchas…

Valor máximo por columna	`max(A)`
Valor mínimo por columna	`min(A)`
Máximo global	`max(max(A))`
Matriz triangular superior / inferior	`triu(A), tril(A)`
Rotación de matriz 90 grados	`rot90(A)`
Inversión de renglones	`flipud(A)`
Inversión de columnas	`fliplr(A)`
Inversión en una dimensión particular	`flipdim(A,dim)`
Corrimiento circular	`circshift(A,corr)`

Operadores

- Muchas muchas!!

Trigonométricas e hiperbólicas

```
sin, cos, tan, asin, acos, atan, atan2
sinh, cosh, tanh, asinh, acosh, atanh
```

Terminación en `d` para manejo en grados, e.g. `sind, acosd`

Exponencial / logaritmicas

```
exp, log, log2, log10, pow2
```

Números complejos

```
real, imag, abs, angle, conj
```

Redondeo

```
round, fix, floor, ceil, sign
```

Funciones de matrices

```
expm, logm, sqrtm, funm
```

Operadores

- Tal vez demasiadas!! ☺

`airy`	Airy functions
`besselj`	Bessel function of first kind
`bessely`	Bessel function of second kind
`besseli`	Modified Bessel function of first kind
`besselk`	Modified Bessel function of second kind
`beta`	Beta function
`ellipj`	Jacobi elliptic functions
`ellipke`	Complete elliptic integrals of first and second kind
`erf`	Error function
`gamma`	Gamma function
`legendre`	Associated Legendre functions

Operador punto

En general un punto (.) antecediendo al operador le indica a Matlab que la operación es elemento por elemento.

Las matrices tienen que tener el mismo tamaño.

```
A =     B =
1 2     5 6
3 4     7 8
```

```
>> B./A
ans =
    5    3
2.333    2
```

```
>> A.^2
ans =
    1    4
    9   16
```

```
>> A.^B
ans =
    1      64
 2187   65536
```

El uso ó desuso indebido del operador (.) es el error de sintaxis más común al empezar a programar en Matlab

Directorios de Matlab

Directorio actual de trabajo (Current directory)

Es la carpeta donde Matlab busca los programas que se desea correr, o salvar los programas nuevos.

El directorio actual se puede cambiar a voluntad.

Directorios del usuario (Set Path)

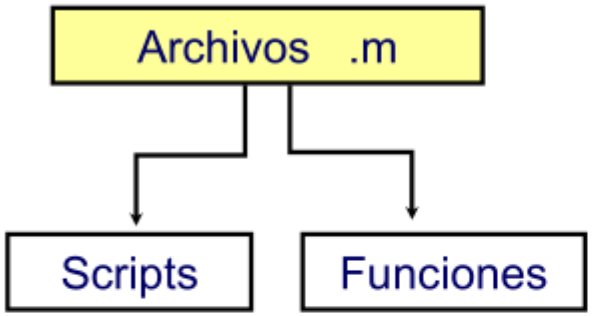
Son carpetas en el disco duro que han sido declaradas previamente donde Matlab busca los programas.

Directorios de Matlab (Program Files)

Son carpetas en el disco duro donde se encuentran los códigos de las funciones internas de Matlab y sus librerías.

Achivos de Matlab

Matlab genera varios tipos de archivos. Los más comunes son:



Archivos texto que contienen programas. Son de dos tipos:

(a) scripts y (b) funciones.

Archivos .mat

Archivos que guardan valores de variables para usarse en el futuro

Archivos .fig

Archivos que contienen la información para reproducir una gráfica en el futuro

Achivos tipo Script

- Sucesión de comandos predefinidos en un archivo.
- Variables globales, que permanecen en memoria.
- Ejemplo: crear un script llamado identidad.m

```
x=-pi:.02:pi;
y= cos(x).^2+sin(x).^2;
```

Archivo del tipo función

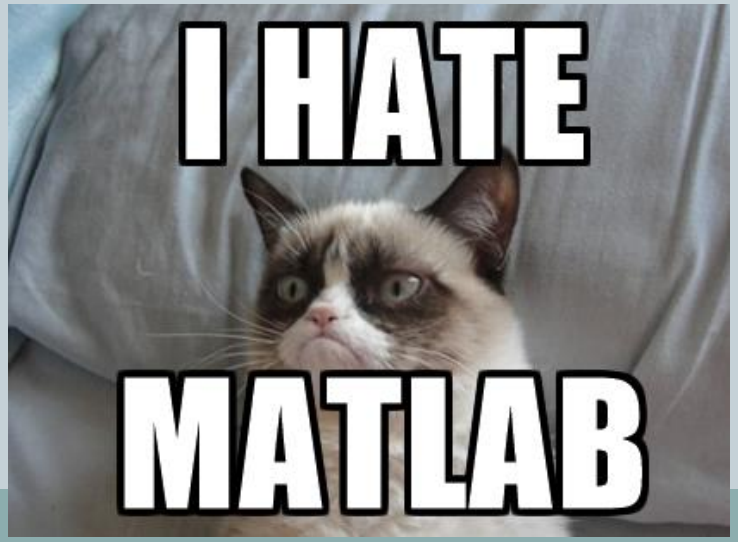
- Son archivos que reciben valores de entrada y generan su salida correspondiente.

function [y1,…,yN] = mifuncion(x1,…,xM)

En donde x1..xM son los parámetros de entrada y y1,…yN son los parámetros de salida.

Archivo del tipo función

- Se recomienda ampliamente salvar el archivo con el nombre de la función.
- Cuidado! Las variables son sólo locales (se borran al terminar la función).



Los legendarios ciclos!

Ciclo if

```
if expresion
   codigo
elseif expresion
   codigo
else
   codigo
end
```

Ciclo for

```
for k = 1:12
   codigo
end
```

Ciclo while

```
while a < tol
   codigo
end
```

Para salir de los ciclos

`continue`	Salir de esa iteración
`break`	Salir del for / while

Los 2 grandes tips para Matlab

- 1.- Lo más importante: VECTORIZAR el código
- 2.- Recordar: PRESASINGAR memoria.

El mantra sagrado: Vectorizar código

- Evitar el uso de ciclos!!
- Por ejemplo, la suma de los primeros N dígitos, se puede escribir mediante el uso de ciclos for, o en su lugar, se puede utilizar alguna función de Matlab.

Además: Pre-asignación de memoria

- Es preferible asignar el espacio que utilizarán ciertas variables en nuestros programas.

Actividad 2

- Crear una función que reciba un valor N como entrada, este valor N corresponderá a un valor entero positivo. La función dará como resultado la suma de 1 hasta N.

1) Programar usando ciclos for.

2) Programar vectorizando el código (help sum).

- Revisar la velocidad de los códigos usando los comandos tic, toc. Reportar códigos y resultados de tiempo para diversos valores de N.

Manejo de texto en Matlab

Se introduce texto usando comillas:

```
>> s = 'tec';
```

Textos se pueden acomodar en arreglos

```
>> h = [s, 'monterrey']
h =
    tec    monterrey
```

De texto a ASCII:

```
>> a = double(s)
a =
   116   101    99
```

De ASCII a texto

```
>> W = char(a)
W =
    tec
```

De variable a texto

```
>> f = 45;
>> g = num2str(f)
g =
    45
```

Conclusiones

- El día de hoy, empezamos a utilizar Matlab.
- Tal vez ahorita parezca algo confuso, pero después de un par de semanas no lo será! (por lo menos no tanto).
- La mejor forma de aprender Matlab: usarlo!
- La actividad será cada dos clases! (sig: 27 de agosto)

Es hora!

- Nos vemos la siguiente semana! Cuídense mucho!!