Laboratorio Introducción a MATLAB Parte 2

Funciones

1. 1. Programas en Matlab

En *Matlab* existen dos tipos de programas: **script** y **function**, y ambos archivos tienen la extension .m. Para ejecutar un programa y/o función, es necesario que *Matlab* esté direccionado a la ubicación correcta. Esto puede hacerse fácilmente ejecutando un programa y luego confirmar que se quiere cambiar de directorio.

- **Script:** Programa estructurado en el cual se pueden ingresar muchas variables a la vez y realizar operaciones con aquellas variables. Estas variables son globales, lo que significa que después de ejecutar el *script*, estas variables quedan almacenadas y pueden verse con el comando *whos*.
- **Function:** Programa al cual se le ingresan variables, realiza operaciones, y devuelve una variable. Son muy útiles para organizar el *script*, ya que se pueden evitar líneas de código al ejecutar una función en vez de ejecutarlas directamente en el *script*. Tienen la siguiente estructura:

$$function[salida1, salida2, ...] = nombre[entrada1, entrada2, ...]$$

Es muy importante que la función se guarde con el mismo nombre que se le dio.

Ejemplo: Queremos resolver la ecuación $3x^2 + 5x + 2 = 0$.
Vamos a crear un *script* con el nombre *eje1.m* (no es nesesario escribir *.m*) Para esto creamos un nuevo archivo de tipo *Script* y escribimos en la ventana del editor:

```
a=3;
b=5;
c=2;
D=b^2 - 4ac % Discriminante
x(1)=(-b+sqrt(D))/(2*a);
x(2)=(-b-sqrt(D))/(2*a);
x  %Pedimos que nos muestre la variable x
```

Para ejecutarlo, se puede escribir el nombre del programa *eje1.m* en la ventana de comando (Command Windows), o simplemente ejecutar el *script* con el botón de play (f5). La desventaja de este tipo de programa es que si se quiere resolver otra ecuación, es necesario modificar el programa.

Ahora vamos a crear una función que resolverá el mismo problema, para esto creamos un nuevo archivo, pero ahora usamos la opción *Function*:

```
function [x]=eje2(a,b,c)
D=b^2 - 4ac % Discriminante
x(1)=(-b+sqrt(D))/(2*a);
x(2)=(-b-sqrt(D))/(2*a);
```

Este archivo **debe** guardarse con el nombre que le dimos (*eje2.m*). Ahora, guarde el archivo. Para utilizar la función haga un nuevo *script* y defina las variables *a,b,c* y finalmente escriba lo siguiente:

```
eje2(a,b,c) %llama a la función con las variables a,b,c
eje2(1,2,3) %resolverá la ecuación con las variables 1,2,3
```

Ahora se pueden resolver ecuaciones de segundo orden llamando a la función *eje2.m*.La ventaja de las funciones es que se pueden ejecutar desde un *script*, lo que ahorrará memoria. Puede también utilizar su función en la ventana de comando.

Ejemplo 1. Escribamos una función de MATLAB tal que, dado un vector $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$,devuelva un vector $F = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}$, tal que $F_i = f(x_i), i = 1, 2, \ldots, n$ siendo:

$$f(x) = \begin{cases} \frac{\log(x)+3}{x} & \text{si } x \neq 0 \\ \frac{1}{2} & \text{si } x = 0 \end{cases} \tag{1}$$

Ahora abramos un nuevo archivo en el editor de texto. ¡Primero asegurémonos de escribir un buen código para nuestra función!

```
function F = mifuncion1(x)
%función para evaluar f(x) = (log(x)+3)/x en cada una de
%las componentes del vector de entrada.
% se crea el vector de ceros con el mismo número de elementos que x
F = zeros(length(x),1);
% ciclo para recorrer el vector y evaluar la función f en los
% elementos de x
for i=1:length(x)
  % probar la condición de que el elemento i-ésimo es distinto de cero
  if x(i) !=0
    F(i) = (1-log(x(i)))/x(i);
  else
    F(i)= 1/2;
  end
end
```

Esta función es la que permite evaluar la función $f(x)$ en cada valor del vector. Ahora guardemos nuestra función en el escritorio (por ahora) o en una carpeta con el nombre *mifuncion1.m*, recuerden que el nombre del archivo y el nombre de la función deben coincidir.

Luego en la ventana de comandos podemos llamar a nuestra función *mifuncion1*. Si queremos probemos con el vector $[0,0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5]$:

```
mifunción(0:0.1:0.5) % evaluar f en [0,0.1,0.2,0.3,0.4,0.5]
whos % no aparecera ninguna variable, ya que todas están dentro de la función
```

Ejemplo 2. Escribamos una script que grafique la función f en 100 puntos equidistantes entre $[1, \pi]$. Abramos un nuevo archivo y escribimos.

```
% Ruteo para gráficar f(x)= f(x) = (log(x)+3)/x entre 1 y pi.
x=linspace(1,pi,100);
fx=mifuncion1(x);
%graficamos
plot(x,fx);
```

Guárdamos la función en el escritorio (o la misma carpeta donde guardamos nuestra función anterior) con el nombre de *plotmifuncion1*, luego en la ventana de comando escribimos:

```
help plotmifuncion1
plotmifuncion
```

1.1. Funciones integradas

Veamos entonces cómo son los programas escritos en matlab, por ejemplo, veamos la función que crea una matriz de Hilbert de `nxn`, la función *hilb(n)*, para esto, escribimos en la línea de comando:

```
edit hilb
```

Este comando abrirá una nueva ventana en el editor, en donde se podrá ver los comentarios de la función, donde indica los ejemplos y una descripción lo que hace la función. Es muy importante al realizar una función dejar anotado que es lo que hace la función y para qué sirve cada variable, es un recordatorio para el futuro.

1.2. Funciones Anónimas

Las funciones anónimas permiten crear funciones simples, sin la necesidad de crear un nuevo archivo, pueden ser definidas en el editor y en la ventana de comandos, utilizando la siguiente sintaxis.

```
NombreF= @(argumentos) expresión
```

NombreF es el nombre que tiene la función para llamarla, *argumentos* son todas las variables separadas por comas (o argumentos) y finalmente la expresión corresponde a la función. por ejemplo escribamos la ecuación de un círculo:

```
circ=@(x,y) x^2 + y^2
```

Una vez realizado esto, podemos escribir en la línea de comandos:

```
circ=@(6,8)
```

```
ans=100
```

También podemos hacer lo siguiente

```
a=3
b=4
circ=@(a,b)
ans=25
```

También podemos hacer gráficas de las funciones anónimas, pero para esto debemos utilizar otro tipo de comando, los llamados función de funciones. En este caso, utilizaremos la función: *fplot(función anonima)*.

```
fplot(@(x) sin(x), [-pi, pi])
```

Debemos mencionar que además que para crear este tipo de funciones existen otros programas que pueden hacer este trabajo de mejor forma.*Mathematica* es un programa especialido para esto. Mientras que *Matlab* se especializa en su capacidad gráfica y poder de procesamiento.

1.3. Introduccion a LaTeX

Es un programa para hacer archivos de alta calidad tipográfica, LaTeXno es un editor de texto, es más bien un lenguaje de programación especializado en la creación de texto.Se utiliza principalmente para la redacción de textos científicos y matemáticos. Su formato de salida por lo general es el PDF o DVI. Es particularmente bueno en escribir ecuaciones matemáticas y dar formato a los textos.

LaTeX, les da formato a los archivos de acuerdo a la clase de documento o \documentclass{book}. En este caso, el programa le da un formato de libro a todo el documento. Otra característica importante de este programa, es que funciona a través de paquetes que permiten nos dan el modo matemático, distintos fonts, tipos de bibliografía, etc. Para utilizarlo, debemos descargar el lenguaje TeX y compilador, cómo **TexMaker** o **TexStudio**, que nos permiten crear los documentos. Otra opción es utilizar un editor online, tal cómo **Overleaf** o **ShareLatex** que sólo funcionan mientras se tenga conexión a internet.