

Práctica 1.- Introducción a Matlab y Números Complejos.

1. Introducción

Matlab es un programa que permite realizar cálculos (numéricos y simbólicos), representaciones gráficas, programación de algoritmos, ...

Al trabajar en Matlab hay que tener en cuenta que el programa distingue entre mayúsculas y minúsculas, es decir, no es lo mismo escribir `>> x = 1` que `>> X = 1`, considerará x y X como dos cosas diferentes.

Al escribir en Matlab, las líneas que terminen en `;` se ejecutarán pero no se mostrará el resultado. Por ejemplo, si escribimos `>> x = 2 * 3 + 1;` no nos mostrará el resultado, aunque si posteriormente escribimos `>> x` nos dará como resultado 7.

El comando `>> clear all` borra todas las variables con las que se ha trabajado hasta ahora. Si escribimos `>> clear x` borrará sólo la variable x .

El comando de ayuda de Matlab es `>> help`. Si escribimos dicho comando seguido del nombre de otro comando o función, nos dará información sobre ello. Por ejemplo, al teclear `>> help cos`, nos proporcionará información sobre cómo funciona la función coseno y una lista de funciones relacionadas.

Utilizamos la orden `>> diary nombredelarchivo.txt` para guardar (en un archivo de texto dentro del directorio actual) lo que vayamos escribiendo en cada sesión. Podemos añadir los comentarios que necesitemos anteponiendo el símbolo `%`, así podremos elaborar nuestros propios apuntes de Matlab. Si queremos dejar de guardar los comandos y salidas en el archivo de texto creado, usaremos el comando `>> diary off`.

Las operaciones básicas en Matlab son:

- La suma y la diferencia: `+` y `-`

Ejemplos:

- $3 + 5$
- $3 - x$

- El producto y el cociente: `*` y `/`

Ejemplos:

- $3 * 2$
- $4/3$

- Las potencias: `^`

Ejemplos:

- $2 \wedge 5$
- $3 \wedge (1/2)$

Además, Matlab tiene predefinidas algunas variables y funciones:

- $\text{sqr}(x)$ nos da la raíz cuadrada de x .
- $\sin(x)$, $\cos(x)$, $\tan(x)$, ... nos dan las razones trigonométricas de x .
- $\exp(x)$ nos da e^x . El número e no está definido previamente, así que para obtenerlo habría que escribir $\exp(1)$.
- $\log(x)$ nos da el logaritmo neperiano de x . Para calcular $\log_a(x)$, hay que aplicar las transformaciones habituales: $\log_a(x) = \frac{\ln(x)}{\ln(a)}$.
- $\text{abs}(x)$ nos da el valor absoluto de x .
- π es el número π .
- Inf es ∞ .
- NaN indica indeterminación. (Not a Number).
- eps nos da el número más pequeño que conoce Matlab.

Ejercicio.- Realiza las siguientes operaciones usando Matlab:

- (a) $\sqrt{3^2 + 1} - \sin(\pi)$
- (b) $e^{2^2+1} + \frac{3}{2}$
- (c) $\log_3(9) + \ln(e) + |\log_{10}(0,1)|$
- (d) $\frac{3^2+5}{\frac{1}{2}+2}$
- (e) $\sqrt[5]{\sqrt{3} + 2}$

En las salidas de Matlab se puede controlar el formato numérico, aunque esto no tiene nada que ver con la precisión con la que se realizan los cálculos de forma interna. Si queremos que se muestren en pantalla los resultados con sólo 5 dígitos, se teclea `>> format short`, y si queremos ver en pantalla los resultados con 15 dígitos, se teclea `>> format long`. Si escribimos `>> format rat`, Matlab busca una aproximación racional del resultado, es decir, lo aproxima por una fracción. Por ejemplo, el número π lo aproxima por la fracción 355/113.

Si se desea obtener el resultado en forma decimal y con un número concreto de decimales, se usa la instrucción `>> vpa(ans,n)`, donde *ans* (o el nombre que se elija) es el nombre de la salida y *n* es el número total de dígitos que usará Matlab para representar su valor.

2. Números Complejos

Realiza por escrito los siguientes ejercicios y comprueba después los resultados usando Matlab.

1. Calcula

a) $\frac{(3-2i)(2+3i)}{3-4i}$

b) i^{2019}

2. Calcula $(1+i)^4$ de las siguientes maneras:

a) En forma polar.

b) Usando el binomio de Newton.

3. Dados $z = 1 + i$ y $w = \frac{9\pi}{4} + i \ln \sqrt{2}$ se pretende hallar el argumento del número complejo z^w cuyo módulo es 1. Para ello seguiremos los siguientes pasos:

a) Calcular $r = |z|$ y $\alpha = \arg(z)$, y comprobar los resultados con Matlab usando los comandos `abs` y `angle`. Recuerda usar el comando `>> sym()`.

b) Hallar la expresión exponencial del módulo de z^w .

c) Hallar k para que $|z^w|$ sea igual a 1.

d) Calcular el argumento de z^w para $k = 1$.

Universidad
de Huelva

Práctica 2.- Ecuaciones en \mathbb{C}

Para resolver ecuaciones con Matlab debemos primero declarar como simbólicas las variables con las que vamos a trabajar. Para ello usaremos el comando `>> syms` seguido de los nombres de las variables separados por espacios.

1. Halla los números complejos z tales que $z^6 - 9z^3 + 8 = 0$.

Comprueba con el comando `>> solve` de Matlab que la resolución de la ecuación es correcta. Puedes usar el comando `>> pretty(ans)` para obtener una expresión más clara de la respuesta.

La solución se asigna por defecto a una variable llamada *ans*, pero si queremos llamar a la solución de otra forma, por ejemplo *sol*, debemos escribir lo siguiente:

```
>> sol = solve(z^6 - 9 * z^3 + 8)
```

2. Resuelve la ecuación $(1 + i)z^3 - 2i = 0$.

Después, utilizando Matlab, da una expresión aproximada con cuatro decimales de la parte real y de la parte imaginaria de cada solución hallada, y comprueba que las tres soluciones son correctas usando el comando `>> subs`.

3. Indica la región del plano que satisface cada una de las siguientes condiciones:

a) $|z - 2| = |z - 1 + i|$,

b) $|z - 2| < 2$,

c) $|z - i| + |z + i| > 4$.

Para comprobar usando Matlab si una igualdad es cierta, se escribe con doble igual (`==`), y la respuesta será `logical 1` si es cierta y `logical 0` si no lo es.

Por ejemplo, para comprobar que $z = i$ forma parte de la región del apartado a), escribiremos `>> abs(i - 2) == abs(i - 1 + i)` y la respuesta debe ser en este caso `logical 1`. Para comprobar que $z = 1 + i$ no forma parte de la región del apartado c), escribiremos `>> abs(1 + i - i) + abs(1 + i + i) > 4` y la respuesta será `logical 0`.

Práctica 3.- Límites, continuidad, asíntotas y gráficas

Para calcular límites con Matlab disponemos de la orden `>> limit`.

Por ejemplo, si queremos calcular

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x}$$

escribiremos `>> limit(sin(x)/x,x,0)`.

Para calcular límites en el infinito, por ejemplo

$$\lim_{x \rightarrow \infty} \frac{x}{e^{x^2}}$$

usaremos `>> limit(x/exp(x^2),x,Inf)`.

Si queremos hacer límites laterales, añadiremos la opción `'right'` o `'left'`.

Por ejemplo, para calcular

$$\lim_{x \rightarrow 0^+} \frac{\sin(x)}{x^2}$$

escribiremos `>> limit(sin(x)/x^2,x,0,'right')`.

Ejercicios.- Realiza por escrito los siguientes ejercicios y comprueba después los resultados usando Matlab.

1. Estudia los siguientes límites:

a) $\lim_{x \rightarrow 1} \frac{x^3 - 1}{(x - 1)^2}$

b) $\lim_{x \rightarrow 0} \frac{6}{4 + e^{-\frac{1}{x}}}$

c) $\lim_{x \rightarrow \infty} (x - \sqrt{x^2 + 2x})$

d) $\lim_{x \rightarrow 0} (1 + 3 \tan^2 x)^{\cot^2 x}$

2. Clasifica las discontinuidades de la función:

$$f(x) = \begin{cases} \frac{\cos \frac{\pi}{x-1}}{1 + e^{\frac{1}{x}}} & \text{si } x \neq 1 \text{ y } x \neq 0, \\ 0 & \text{si } x = 0. \end{cases}$$

3. Determina el dominio, el signo y calcula las asíntotas de las siguientes funciones reales:

a) $f(x) = \frac{2x}{\sqrt{x^2 - 1}}$.

b) $f(x) = \ln\left(\frac{x+1}{x-1}\right)$.

c) $f(x) = \frac{1}{e^x - 1}$.

Para dibujar funciones en Matlab podemos usar la orden `>> ezplot`. Por ejemplo, si queremos ver en pantalla la función

$$f(x) = \frac{1}{e^x - 1}$$

con valores de x comprendidos entre -25 y 25 , escribiremos

```
>> ezplot(1/(exp(x) - 1), [-25, 25]).
```

Recordamos que para sustituir el valor $x = x_0$ en una función $y = f(x)$, podemos usar la orden `>> subs(y, x0)`.

Universidad
de Huelva

Práctica 4.- Archivos .m

En Matlab se pueden crear archivos de extensión .m para definir nuevas funciones que se podrán usar como se usa cualquiera de las que ya estaban definidas en Matlab.

Para crear este tipo de archivos podemos usar el botón *New* y, en el menú que despliega, elegir *Function*. De esta manera, aparecerá una ventana en la que escribiremos las instrucciones de nuestra nueva función que deberemos “guardar como” con el mismo nombre que le demos a la función creada (**nombre.m**). Este nombre debe empezar con una letra y no debe coincidir con el nombre de ninguna de las funciones de las que ya dispone Matlab.

La primera línea del archivo de instrucciones es de la forma

function[y,z,...] = **nombre**(a,b,...)

donde **nombre** es el que le hayamos dado al archivo .m; entre corchetes, porque es opcional, aparecen la o las variables de salida (*outputs*) y entre paréntesis aparecen la o las variables de entrada (*inputs*). Si en la siguiente línea incluimos un comentario precedido de %, ésa será la explicación que dé Matlab al solicitar ayuda con el comando >> **help** seguido del nombre de la función.

Para usar la función creada, el directorio actual (o *Current Folder*) debe ser la carpeta en la que se haya guardado el archivo de extensión .m creado. Para que al ejecutar nuestro archivo de función no aparezca, además de las salidas que hayamos programado, la respuesta **ans**, debemos poner punto y coma al final de la orden que escribimos para usar la función.

La sintaxis usada para **if**, **for** y **while** es la siguiente:

◇ **if** *expresión lógica*

instrucciones

elseif *expresión lógica*

instrucciones

elseif *expresión lógica*

instrucciones

...

else

instrucciones

end

◇ **for** *índice = inicio : incremento: final*

instrucciones

end

◇ **while** *expresión lógica*

instrucciones

end

Ejemplo 1.- Dados los coeficientes de una ecuación de segundo grado, vamos a crear un archivo de función que nos diga el número de soluciones reales que tiene la ecuación, y cuáles son dichas soluciones.

```
function[x1,x2] = ecu2(a,b,c)
%Para resolver la ecuación de coeficientes a, b y c escribiremos la orden >> ecu2(a,b,c);
Delta = b^2 - 4 * a * c; %Se calcula el discriminante
if Delta < 0
    disp('No hay soluciones reales ')
elseif Delta > 0
    x1 = (-b - sqrt(Delta))/(2 * a);
    x2 = (-b + sqrt(Delta))/(2 * a);
    disp('Hay dos soluciones reales '), x1, x2
else
    x1 = -b/(2 * a);
    x2 = -b/(2 * a);
    disp('Solo hay una solucion real '), x1, x2
end
end
```

1. Resuelve las siguientes ecuaciones de segundo grado mediante la función ecu2:

(1.a) $x^2 - 4x - 5 = 0$

(1.b) $x^2 - x + 1 = 0$

(1.c) $x^2 + 6x + 9 = 0$

Ejemplo 2.- Vamos a crear una función que corresponda a

$$f(x) = \begin{cases} \frac{e^x - 1}{x} & \text{si } x \neq 0 \\ 1 & \text{si } x = 0 \end{cases}$$

```
function[y] = fun(x)
if x == 0
    y = 1;
else
    y = (exp(x) - 1)/x;
end
end
```

Ya podemos usar esta función como cualquier otra de Matlab simplemente llamándola por su nombre. Así, por ejemplo, si escribimos `>> fun(2)`, nos dará el valor que toma la función para $x = 2$.

Para usarla como input de otra función, debemos escribir su nombre como cadena de caracteres, es decir, `'fun'`. Por ejemplo, `>> ezplot('fun', [-2, 5])` dará su representación gráfica en el intervalo indicado.

Ejemplo 3.- Basado en el Teorema de Bolzano, el *Método de la Bisección* aproxima una raíz de una función localizada en un intervalo. Vamos a crear un archivo llamado `bisection.m`:

```
function[c,err,yc,iter] = bisection(f,a,b,delta)
% Para usar esta función bisection primero tenemos que haber definido la f,
% por ejemplo: >> f1(x) = sin(x) - 1/2
% Escribiremos las entradas de esta forma: >> bisection(f1,0,2,0.001);
% siendo en este caso el intervalo [a,b] = [0,2]
% y delta=0.001, la precisión, es decir, tres cifras decimales exactas.
% c es el cero buscado
% yc será el valor de f en c que debe ser muy próximo a 0
% err es el error de la aproximación de c
% iter es el número de iteraciones realizadas para dar c.
ya = feval(f,a);
yb = feval(f,b);
if ya*yb > 0
    disp('No se puede aplicar el Teorema de Bolzano en este intervalo.')
    return
end
err = b - a;
iter = 0;
while err >= delta
    c = (a + b)/2;
    yc = feval(f,c);
    if yc == 0
        a = c;
        b = c;
    elseif yb*yc > 0
        b = c;
        yb = yc;
    else
        a = c;
        ya = yc;
    end
    err = b - a;
    iter = iter + 1;
end
disp('El cero es '),c
err = abs(b - a);
disp('El error es menor que '),err
yc = vpa(feval(f,c),10);
disp('El valor de la funcion en c es '),yc
disp('El numero de iteraciones ha sido '),iter
end
```

- Halla una aproximación de la raíz de la función $f(x) = x - 3^{-x}$ con un error menor que 0.01, determinando previamente, mediante un gráfico, el intervalo en el que se localiza dicha raíz.
- Aplica el método de bisección para calcular, con un error menor que 0.01, un cero de la función $g(x) = x^2 - \sin(x + 0,15)$.

PRÁCTICA 2

%Ejercicio 1 explicado de dos formas

%Primera

syms z

solve(z^6 - 9 * z^3 + 8)

ans =

1

2

- 1 - 3^(1/2)*1i

- 1 + 3^(1/2)*1i

- (3^(1/2)*1i)/2 - 1/2

(3^(1/2)*1i)/2 - 1/2

pretty(ans)

/ 1 \

| |

| 2 |

| |

| - 1 - sqrt(3) 1i |

| |

| - 1 + sqrt(3) 1i |

| |

| sqrt(3) 1i 1 |

| - ----- - |

| 2 2 |

| |

| sqrt(3) 1i 1 |

| ----- - |

$$\sqrt{\quad^2 \quad^2} /$$

%Segunda forma usando roots

p=[1 0 0 -9 0 0 8]

p =

1 0 0 -9 0 0 8

roots(p)

ans =

-1.0000 + 1.7321i

-1.0000 - 1.7321i

2.0000 + 0.0000i

-0.5000 + 0.8660i

-0.5000 - 0.8660i

1.0000 + 0.0000i

%Segundo ejercicio

s=[(1+i) 0 0 2*i]

s =

1.0000 + 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 2.0000i

s=[(1+i) 0 0 -2*i]

s =

1.0000 + 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 - 2.0000i

roots(s)

ans =

-0.7937 + 0.7937i

-0.2905 - 1.0842i

1.0842 + 0.2905i

%tercer ejercicio

%El profe nos comentó que solamente hay que decir que region geometrica es cada cosa. El primero

% es una mediatriz, el segundo una circum y el ultimo una elipse

diary off

PRÁCTICA 3

%ejercicio1

syms x

%apartado a)

limit((x^3-1)/(x-1)^2,x,1)

ans =

NaN

limit((x^3-1)/(x-1)^2,x,1,'left')

ans =

-Inf

limit((x^3-1)/(x-1)^2,x,1,'right')

ans =

Inf

factor(x^3-1)

ans =

[x - 1, x^2 + x + 1]

%apartado b)

limit(6/(4+exp(-1/x))),x,1,)

limit(6/(4+exp(-1/x))),x,1,)

{ Error: Invalid expression. When calling a function or indexing a variable, use parentheses.
Otherwise, check for mismatched

delimiters.

}

limit(6/(4+exp(-1/x))),x,0)

limit(6/(4+exp(-1/x))),x,0)

{ Error: Invalid expression. When calling a function or indexing a variable, use parentheses.
Otherwise, check for mismatched

delimiters.

}

limit(6/(4+exp(-1/x))),x,0

ans =

NaN

x =

x

ans =

0

limit(6/(4+exp(-1/x))),x,0, 'left')

limit(6/(4+exp(-1/x))),x,0, 'left')

{ Error: Invalid expression. When calling a function or indexing a variable, use parentheses.
Otherwise, check for mismatched

delimiters.

}

limit(6/(4+exp(-1/x))),x,0, 'left')

ans =

0

limit(6/(4+exp(-1/x))),x,0, 'right')

ans =

3/2

%apartado c)

```
limit(x-sqrt(x^2+2*x),x,inf)
```

ans =

-1

%apartado d)

```
limit((1+3*(tan(x))^2)^(1/tan(x)^2),x,0)
```

ans =

exp(3)

%Ejercicio 2

```
limit(cos(pi/(x-1))/(1+exp(1/x)),x,0)
```

ans =

NaN

```
limit(cos(pi/(x-1))/(1+exp(1/x)),x,0 'left')
```

```
limit(cos(pi/(x-1))/(1+exp(1/x)),x,0 'left')
```

{ Error: Invalid expression. Check for missing multiplication operator, missing or unbalanced delimiters, or other syntax error. To

construct matrices, use brackets instead of parentheses.

}

```
limit(cos(pi/(x-1))/(1+exp(1/x)),x,0, 'left')
```

ans =

-1

limit(cos(pi/(x-1))/(1+exp(1/x)),x,0, 'right')

ans =

0

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1)

ans =

NaN

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1, 'left')

ans =

NaN

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1, 'right')

ans =

NaN

%Ejercicio 3

%apartado a

%aprtadoa)*

%apartado a)** soy inutil

f=2*x/sqrt(x^2-1)

f =

(2*x)/(x^2 - 1)^(1/2)

subs(x^2-1,x,-2)

ans =

3

subs(x^2-1,x,0)

ans =

-1

subs(x^2-1,x,2)

ans =

3

subs(f,x,-2)

ans =

-(4*3^(1/2))/3

$$-(4*3^{(1/2)})/3$$

ans =

$$-2.3094$$

subs(f,x,2)

ans =

$$(4*3^{(1/2)})/3$$

$$(4*3^{(1/2)})/3$$

ans =

$$2.3094$$

limit(f,x,-1, 'left')

ans =

$$-\text{Inf}$$

limit(f,x,-1, 'right')

ans =

$$\text{Inf} \cdot 1i$$

```
limit(f,x,-inf)
```

```
ans =
```

```
-2
```

```
limit(f,x,inf)
```

```
ans =
```

```
2
```

```
ezplot(f,[-3,3])
```

```
fplot(f,[-3,3])
```

```
%estas dos cosas son pa ver las funciones
```

```
%apartado b)
```

```
f=log((x+1)/(x-1))
```

```
f =
```

```
log((x + 1)/(x - 1))
```

```
subs (log((x+1)/(x-1)),x,0)
```

```
ans =
```

```
pi*1i
```

```
subs ((x+1)/(x-1),x,0)
```

```
ans =
```

-1

subs ((x+1)/(x-1),x,2)

ans =

3

subs (f,x,2)

ans =

log(3)

subs (f,x,-2)

ans =

-log(3)

subs (f,x,-1,'left')

```
{ Error using <a
href="matlab:matlab.internal.language.introspective.errorDocCallback('sym/subs')"
style="font-weight:bold">sym/subs</a>
```

Too many input arguments.

```
}
```

limit (f,x,-1,'left')

ans =

-Inf

```
limit (f,x,-1,'right')
```

```
ans =
```

```
- Inf + pi*1i
```

```
limit (f,x,1,'right')
```

```
ans =
```

```
Inf
```

```
fplot(f,[-3,3])
```

```
limit (f,x,inf)
```

```
ans =
```

```
0
```

PRACTICA 4

Archivos .m

edit

primera linea del archivo es function [y,z,...] = nombre (a,b,...)

IF, FOR, WHILE.

* IF		* FOR → índice = inicio : incremento : final
elseif		instrucciones
elseif		end.
else		
END		* WHILE expresion logica
		instrucciones.
		END.

TEOREMA BOLZANO

$F(b) > 0 \rightarrow 0$ corta el eje
 $F(b) < 0$ y es 0.

↓
existe una raíz.

↓
Obtener Numero

↓
metodo de Bisección.

Ejemplo 1.

1.a) $\rightarrow x^2 - 4x - 5 = 0$

ecuz(1,-4,-5) \rightarrow Dos soluciones reales
 $\rightarrow = 5$.

1.b). $x^2 - x + 1 = 0$. \rightarrow No hay soluciones reales.

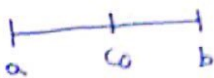
1.c). $x^2 + 6x + 9 = 0 \rightarrow x = 3$.

Ejemplo 2

fun(z)

ezplot('fun', [-2, 5]).

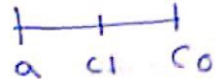
② $f[a, b]$ $f(a) < 0$ $f(b) > 0$

$I_0 \rightarrow c_0 = \frac{a+b}{2}$  $\rightarrow E \leq \frac{b-a}{2}$

$f(c_0) = 0 \rightarrow c_0$ es la raíz.

Supongamos $f(c_0) > 0$

$[a, c_0] \rightarrow$ cambio signo

$I_1 \rightarrow c_1 = \frac{a+c_0}{2}$  $E \leq \frac{b-a}{4} = \frac{b-a}{2^2}$

$f(c_1) = 0 \rightarrow c_1$ es la solución.

$f(c_1) > 0 \rightarrow [a, c_1] \dots$

miramos con `ezplot(fuccion)` donde corta los ejes. en un. un
diferente.

después ejecutamos `bisectan('fexer2', 0, 1, 0.001);`

error $\frac{b-a}{2^n}$