

PRÁCTICA Temas 1 y 2

1. Introducción a MATLAB

Al abrir MATLAB nos aparecen en pantalla varias pestañas, barras y ventanas. La distribución de ventanas en la pantalla depende de la configuración elegida por el usuario. Si, con la pestaña *HOME* activa, pinchamos en el icono *Layout* y elegimos la opción *Default*, aparecerá en pantalla la distribución por defecto que describiremos a continuación.

En ella podemos ver las siguientes ventanas:

- *Current Folder*: Indica el directorio de trabajo y nos muestra los ficheros en él contenidos.
- *Command Window*: Es la que utilizamos para escribir las instrucciones a continuación del *prompt* `>>`.
- *Workspace*: Muestra las variables en uso con información sobre ellas.
- *Command History*: Almacena un historial de las órdenes escritas en la ventana de comandos. Nos permite, entre otras cosas, repetir o utilizar órdenes ya ejecutadas.

Pinchando sobre el icono *Help* de la barra de herramientas, podemos acceder a diversos menús de ayuda. Si queremos obtener información sobre un comando concreto, basta con escribirlo en la ventana de comandos precedido de la orden **help** o también de la orden **doc**. Por ejemplo:

```
>> help sqrt
>> doc sqrt
```

1.1. Trabajo en la ventana de comandos

Para realizar cálculos numéricos elementales con MATLAB, basta con conocer la sintaxis de las distintas operaciones:

Suma	Resta	Multiplicación	División	Potenciación
+	-	*	/	^

y tener en cuenta que la jerarquía de dichas operaciones se estructura con una prelación determinado. Así, en primer lugar se efectúan los paréntesis, luego las potencias, después productos y cocientes, y, finalmente, sumas y restas. Dentro de un mismo nivel, las operaciones se realizan en el orden en el que están escritas, de izquierda a derecha. Lógicamente, si deseamos alterar el orden de ejecución de las operaciones deberemos utilizar paréntesis.

Otras cosas a tener en cuenta son:

- Las teclas \uparrow y \downarrow permiten recuperar líneas de comando escritas con anterioridad, mientras que con \rightarrow y \leftarrow podemos movernos a derecha e izquierda en una línea.
- También es posible recuperar una orden escrita anteriormente, desde la ventana *Command History*. Para ello podemos pinchar dos veces sobre ella o arrastlarla con el ratón hasta la ventana de comandos.
- Pueden escribirse varias órdenes en una misma línea separándolas con los símbolos “;” ó “;”. Éste último, detrás de una orden, evita que el resultado se muestre por pantalla.
- El comando **clc** limpia la ventana de comandos, pero eso no afecta a los datos almacenados en memoria. Tiene una finalidad meramente estética.
- El símbolo **%** es utilizado para insertar comentarios. Todo lo que se escriba a su derecha, en esa línea, es omitido por MATLAB.

1.1.1. Formatos

Cuando MATLAB presenta los resultados, elige por defecto un formato con un máximo de 3 dígitos para la parte entera y 4 para la parte decimal (formato corto). Si el tipo de cálculos que deseamos realizar requieren de un manejo de más decimales o queremos que los resultados se muestren con otro tipo de formato, podemos hacer uso del comando **format**:

Orden	Tipo de formato
format short	Los valores se muestran con un máximo de 3 dígitos para la parte entera y 4 para la decimal; si el número es mayor, se expresa con notación exponencial.
format long	Los valores se muestran con un máximo de 2 dígitos para la parte entera y 15 para la decimal; si el número es mayor, se expresa con notación exponencial.
format rat	Aproximación racional.

El formato utilizado afecta sólo a cómo se muestran los datos por en la pantalla, no a cómo MATLAB los calcula o los guarda.

Ejercicio 1.1

1. Hallar el valor de $5^2 - 17 \cdot 2 + 7/5 - \sqrt{12}$. (Sol: -11.0641)

2. Observar la diferencia entre las siguientes operaciones:

$$3^2 - 5 \left(2 - \frac{3}{4} \cdot 7 \right); \quad 3^2 - 5 \cdot 2 - \frac{3}{4 \cdot 7}$$

(Sol: 25.2500, -1.1071)

3. Hallar el valor de $7^2 - 17 \cdot 2 + 7/5 - \sqrt{15}$, recuperando la orden del apartado 1 con la tecla \uparrow y modificando los números correspondientes. (Sol: 12.5270)

4. Analizar los resultados de las siguientes operaciones:

```
>> pi, format long, pi, format rat, pi, format short
>> 10*pi, 100*pi, 1000*pi, 2e3, 2e-3
```

1.2. Variables

Una variable es una estructura a la que se asigna un contenido para ser utilizado posteriormente. El contenido de una variable se asigna mediante el símbolo `=` y será permanente hasta que sea borrada la variable o se realice una reasignación. El nombre de la variable siempre debe ir a la izquierda del símbolo `=` y su contenido a la derecha.

1.2.1. Reglas para nombrar variables

Para nombrar variables debemos seguir las siguientes reglas:

- MATLAB distingue entre letras mayúsculas y minúsculas a todos los efectos.
- El nombre de una variable puede contener un máximo de 31 caracteres, ignorándose los posteriores.
- El nombre de una variable debe empezar necesariamente por una letra, aunque puede contener letras, números y el guión bajo. Nunca puede contener operadores (+, *, etc.), espacios en blanco ni signos de puntuación.
- No deben nombrarse variables con expresiones que tengan un significado específico de funciones o variables predefinidas en MATLAB: **pi**, **cos**, **log**, etc. puesto que perderemos su finalidad hasta que dicha variable sea borrada.

Ejercicio 1.2

1. Ejecutar la orden `34=b` y observar lo que ocurre.
2. Almacenar en las variables `x`, `z` y `area` los valores 3, 5 y 27, respectivamente.
3. Calcular el valor `x+area`.
4. Asignar a los nombres `a1` y `1a` los valores 8 y 88 y observar lo que ocurre.
5. Obtener el valor de $5+\sqrt{2}$ y ejecutar a continuación la orden **`ans`**. Observar que en **`ans`** se almacena el resultado de la última orden ejecutada.

Nótese que en la ventana *Workspace* se almacenan las variables que vamos creando.

1.2.2. Cómo borrar variables

La orden **`clear`** (o **`clear all`**) se utiliza para borrar todas las variables en uso. Suele ser útil su empleo al comenzar de cero un ejercicio o cuando en ocasiones aparecen errores que, en principio, no tendrían que hacerlo. Si a la orden se le añade una lista de variables (separadas por espacios en blanco) sólo se borrarán esas.

Ejercicio 1.3

1. Limpiar la ventana de comandos con la orden **`clc`**. Obsérvese que siguen en memoria todas las variables en uso.
2. Borrar las variables `x` y `area`.
3. Bórrense el resto de las variables.

1.2.3. Algunas variables predefinidas en MATLAB

Nombre	Significado
ans	Almacena el último resultado no asignado a una variable
eps	Epsilon de la máquina
pi	π
i ó j	Unidad imaginaria
inf	∞
NaN	No es un número (Not a Number)
date	Fecha

eps es el número positivo más pequeño que sumado a 1 genera un número mayor que 1 en el ordenador (en un PC, **eps**=2.220446049250313e-016).

NaN representa una expresión indeterminada, como puede verse si escribimos, por ejemplo:
`>> (2-2)/(3-3).`

1.2.4. Funciones matemáticas predefinidas en MATLAB

La orden **`doc elfun`** nos permite obtener un listado completo todas estas funciones. Algunos ejemplos de ellas son:

Nombre	Significado
abs(x)	valor absoluto de x
sin(x)	seno de x
cos(x)	coseno de x
tan(x)	tangente de x
asin(x)	arcoseno de x
acos(x)	arcocoseno de x
atan(x)	arcotangente de x
exp(x)	exponencial de x
log(x)	logaritmo neperiano de x
sqrt(x)	raíz cuadrada de x
nthroot(x,n)	raíz real n-ésima de x

Ejercicio 1.4

1. Determinar el valor de $5+\pi$.
2. Definir una variable con el nombre **`pi`** y almacenar en ella el valor 6.
3. Calcúlese $5+\pi$. Nótese que hemos perdido el valor de la variable predefinida en MATLAB.
4. Borrar la variable `pi` y obsérvese que hemos recuperado su valor original.
5. Calcular los valores de $\sin(\pi/2)$ y e^4 .
6. Obtener el valor de $\sqrt{2}$ de dos formas distintas (mediante el comando **`sqrt`** y como $2^{1/2}$).

1.2.5. Constantes y variables simbólicas

Los cálculos en MATLAB se realizan, por defecto, en formato numérico. Si efectuamos operaciones como $1/2 + 1/3$ ó $(\pi^2 - 1)/(\pi - 1)$, nos devuelve los valores aproximados, 0.8333 y 4.1416, en vez de los resultados exactos $5/6$ y $\pi + 1$, respectivamente.

Por otra parte, hay cálculos habituales en Matemáticas y que no se pueden realizar con las órdenes de MATLAB estudiadas hasta el momento, como por ejemplo:

$$\begin{aligned} \blacksquare (a+b)(a-b) &= a^2 - b^2 \\ \blacksquare \int 2x \, dx &= x^2 \end{aligned}$$

ya que MATLAB no trabaja con variables que no tengan valores numéricos asignados.

Para solventar este tipo de inconvenientes, podemos utilizar la herramienta *cálculo simbólico* de MATLAB. Denominaremos *expresiones, constantes y variables simbólicas* a aquellas que intervengan en las manipulaciones de tipo simbólico.

Para declarar variables simbólicas utilizaremos la orden **syms**, escribiendo a continuación, y separadas por espacios en blanco, dichas variables.

Toda expresión en la que aparezcan variables simbólicas es, automáticamente, considerada también como tal.

```
>> syms y z % Crea las variables simbólicas y, z
>> g=z^2+y % Crea la expresión simbólica g
```

Si en una expresión simbólica queremos sustituir una variable por otra, o por una constante, utilizaremos la orden **subs**. Si la sustitución afecta a más de una variable las escribiremos entre llaves y separadas por comas:

```
>> syms t
>> f=subs(g,{y,z},{t,3}) % Sustituimos, en g, y por t y z por 3
>> h=subs(f,t,2)
```

Para convertir una variable numérica en simbólica se utiliza el comando **sym**:

```
>> a=sym(pi) % Almacena en a la constante pi como simbólica
```

En sentido inverso, la orden **double** convierte a formato numérico el contenido de una variable simbólica.

```
>> n=double(a) % Aproxima el valor numérico de pi y lo almacena en n
```

Por último, la orden **vpa(s,d)** facilita el valor numérico de s, utilizando d dígitos:

```
>> s=sym(sqrt(2))
>> s=vpa(s,6)
```

Ejercicio 1.5

1. Definir la expresión simbólica $f = x^2 + y^2$.
2. Construir las constantes simbólicas $a = 1/4$, y $b = \sqrt{2}$.
3. Sustituir en f , x e y por a y b .

1.3. Archivos de órdenes

Cuando queremos guardar y/o ejecutar un gran número de instrucciones de MATLAB no resulta operativo trabajar directamente en la ventana de comandos. Vamos a crear unos *ficheros de comandos* o *ficheros de programa* en MATLAB que se caracterizan por tener la extensión **.m**. Para los nombres de ficheros se siguen las mismas reglas que para nombrar variables.

Ejercicio 1.6

1. Seleccionar el primer icono de la barra de herramientas (o también: *New --> Script*).
2. En la ventana del editor de MATLAB que se ha abierto, generamos un fichero con las siguientes órdenes:


```
% Sumar y multiplicación dos números
x=5
y=7
S=x+y % Ésta es la suma
```

3. Podemos ejecutar sólo algunas de las órdenes escritas (incluso antes de salvar el fichero) seleccionándolas con el ratón y pulsando a continuación la tecla **F9**. Obsérvese que las órdenes se ejecutan en la ventana de comandos.
4. Para guardar el fichero, en el directorio elegido para trabajar, pinchar en el icono Save. Asignarle como nombre **prueba.m**.
5. Volviendo nuevamente al editor, añadir al fichero la orden:

```
P=x*y % Éste es el producto
```

Obsérvese que detrás del nombre del fichero, que está bajo la barra de herramientas, aparece un asterisco. Nos indica que lo que estamos viendo en pantalla no es la versión guardada del fichero. Desaparecerá en cuanto volvamos a guardar, tal y como explicamos anteriormente. Es importante asegurarse de que siempre que ejecutamos un fichero, lo hagamos sobre la versión actualizada.

6. Comprobar que el fichero **prueba.m** aparece en la ventana Current Folder.
7. Una vez salvado el fichero, ejecutamos secuencialmente todas las órdenes que contiene escribiendo en la ventana de comandos su nombre (sin la extensión) y pulsando a continuación **enter**. También podemos hacerlo pinchando sobre el icono Run (en este caso guarda y ejecuta simultáneamente)
8. Nótese que los comentarios (precedidos de **%**) no salen en pantalla.
9. Escribir en la ventana de comandos **help prueba** y aparecerán, a modo de ayuda, los comentarios que hayamos escrito antes de la primera línea ejecutable.

En ocasiones resulta cómodo integrar el editor, como una ventana más, dentro del marco principal de MATLAB. Para ello, basta con pinchar en el icono  de la parte superior derecha del editor y elegir Dock Editor o Undock Editor según queramos integrarle o extraerle de dicho marco principal.

2. Números complejos

En MATLAB la unidad imaginaria se representa con las letras **i** ó **j**. Las operaciones básicas con números complejos se realizan igual que con los números reales. Siempre que la parte imaginaria de un número complejo contenga alguna función u operación, deberá escribirse el símbolo de producto, *****, entre la parte imaginaria y la unidad imaginaria.

Otras funciones útiles para operar con complejos son las siguientes:

Orden	Salida
<code>real(z)</code>	Parte real de z .
<code>imag(z)</code>	Parte imaginaria de z .
<code>abs(z)</code>	Módulo de z .
<code>conj(z)</code>	Conjugado de z .
<code>angle(z)</code>	Argumento que se encuentra en el intervalo $(-\pi, \pi]$.

Ejercicio 2.1

1. Expresar en forma binómica los siguientes números complejos:

$$z = \frac{1 - e^{\pi i/2}}{1 + e^{\pi i/2}} \quad w = e^{\pi i}(1 - e^{-\pi i/3})$$

2. Calcular el módulo, el argumento, la parte real y la parte imaginaria de $u = \frac{z + \bar{w}}{2i} + \frac{1}{2}$.

(Para trabajar con el número π de forma exacta, podemos utilizarle como constante simbólica)

Ejercicio 2.2

Dados los números complejos $z = 1 + i$ y $w = 1 - \sqrt{3}i$ se pide:

1. Calcular sus módulos y sus argumentos (los argumentos como constantes simbólicas) para poder expresarlos en sus distintas formas (módulo-argumental, trigonométrica y exponencial).
2. Utilizar el apartado anterior para calcular en forma exponencial $(1 + i)^4(1 - \sqrt{3}i)^2$.

Ejercicio 2.3

Dado el número complejo $z = -i$, se pide:

1. Calcular su módulo y su argumento.
2. Sabiendo que, si $z = r \cdot e^{i\theta} \neq 0$ y $n \in \mathbb{N}$, las n raíces n -ésimas de z son:

$$w_k = \sqrt[n]{r} \cdot e^{i \frac{\theta + 2k\pi}{n}} \text{ para } k \in \{0, \dots, n-1\}$$

calcular las tres raíces cúbicas, w_0 , w_1 y w_2 de $-i$.

3. Ejecutar las órdenes:

```
>> plot([w0,w1,w2,w0], 'k'), hold on, plot([w0,w1,w2], '*r')
>> ezplot('x^2+y^2-1', [-1,1, -1,1])
```

3. Matrices

3.1. Vectores y matrices

El tipo básico de dato con el que trabaja MATLAB es la matriz. Los escalares son considerados como matrices 1×1 , los vectores de n componentes como matrices $1 \times n$, etc.

Una matriz se define entre corchetes, introduciendo los elementos de cada fila separados por comas o espacios, y cambiando de fila mediante un punto y coma. Por ejemplo, podemos hacer:

```
>> v=[1 3 pi 1/3]
>> A=[-1,0,1;2 -1 0;3 0 -3]
```

En el caso de los vectores, se pueden definir de manera específica cuando sus componentes estén equiespaciadas, esto es, cuando la diferencia entre dos componentes consecutivas del vector sea constante:

Orden	Salida
<code>[a : h : b]</code>	Vector $(a, a+h, a+2h, \dots, a+nh)$, es decir, sus componentes van desde a hasta b , de h en h , sin sobrepasar el valor b . En este caso, los corchetes pueden sustituirse por paréntesis o incluso eliminarse. El incremento h puede ser omitido y, en ese caso, se toma como 1.
<code>linspace(a,b,n)</code>	Vector de n componentes cuyas coordenadas son los puntos de una partición uniforme del intervalo $[a, b]$.

```
>> u1=[1:0.3:2]
>> u2=(1:-0.4:-0.8)
>> u3=linspace(1,2,5)
```

Una vez definidos un vector o una matriz, podemos acceder a sus elementos o submatrices con las siguientes órdenes:

Orden	Salida
<code>v(i)</code>	Coordenada i del vector v .
<code>A(i,j)</code>	Elemento (i,j) de la matriz A .
<code>A(:,j)</code>	Columna j de la matriz A .
<code>A(:,end)</code>	Última columna de la matriz A .
<code>A(i,:)</code>	Fila i de la matriz A .
<code>A(end,:)</code>	Última fila de la matriz A .
<code>A(v,w)</code>	Submatriz de A que contiene las filas y columnas indicadas en las coordenadas de v y w , respectivamente.
<code>A(i,:)=[]</code>	Elimina la fila i de la matriz A .
<code>A(:,j)=[]</code>	Elimina la columna j de la matriz A .

Nótese que `[]` representa la matriz vacía (*empty matrix*).

```
>> v(2)
>> v([2:3])
>> A(1,2)
>> A(:,2)
>> A([2,1],2:3)
```

Pueden definirse ciertos tipos de matriz con las siguientes órdenes:

Orden	Salida
<code>ones(n)</code>	Matriz cuadrada $n \times n$ de unos.
<code>ones(m,n)</code>	Matriz $m \times n$ de unos.
<code>zeros(n)</code>	Matriz cuadrada $n \times n$ de ceros.
<code>zeros(m,n)</code>	Matriz $m \times n$ de ceros.
<code>eye(n)</code>	Matriz identidad $n \times n$.
<code>eye(m,n)</code>	Matriz $m \times n$ con unos en la diagonal principal y ceros en el resto.
<code>diag(v)</code>	Matriz diagonal, con las componentes de v en la diagonal principal.
<code>rand(m,n)</code>	Matriz $m \times n$ con números comprendidos entre 0 y 1 generados aleatoriamente.

Ejercicio 3.1

Construir una matriz A , de orden 5×5 , cuyos elementos sean todos 1. Modificar la tercera fila, de manera que todos sus elementos sean 20.

```
>> A=ones(5)
>> A(3,:)=20*A(3,:)
```

3.1.1. Matrices por bloques

Podemos definir matrices por bloques o “matrices de matrices”, aquellas cuyos elementos sean, a su vez, matrices. Veamos algún ejemplo:

```
>> A=ones(3), B=eye(3,2), C=zeros(2,3)
>> D=[A,B]
>> E=[C;A]
>> F=[D,A;A,D;2:9]
```

3.1.2. Operaciones con matrices

Sean $A = (a_{ij})$ y $B = (b_{ij})$ dos matrices con las dimensiones adecuadas y λ un escalar. Las operaciones habituales se efectúan con las siguientes órdenes:

Operación	Resultado
$A+B$	Suma A y B .
$A-B$	Suma a A el opuesto de B (Resta B de A).
$A*B$	Producto de A por B .
A/B	Calcula AB^{-1} .
$A \setminus B$	Calcula $A^{-1}B$.
$\lambda * A$	Multiplica todos los elementos de A por λ .
A^n	Eleva la matriz A al entero n (Multiplica A por sí misma n veces).
$A.'$	Calcula la traspuesta de A .
A'	Calcula la traspuesta de la conjugada de A .

Además de estas operaciones, en MATLAB se definen otras que llamaremos **operaciones elemento a elemento**:

Operación	Resultado
$\lambda + A$	Suma a cada elemento de A el escalar λ .
$A.*B$	Matriz que en la posición (i,j) contiene el producto $a_{ij}b_{ij}$.
$A./B$	Matriz que en la posición (i,j) contiene el cociente a_{ij}/b_{ij} .
$\lambda./A$	Matriz que en la posición (i,j) contiene el cociente λ/a_{ij} .
$A.^{\lambda}$	Matriz que en la posición (i,j) contiene $(a_{ij})^{\lambda}$.
$A.^B$	Matriz que en la posición (i,j) contiene $(a_{ij})^{b_{ij}}$.
$\lambda.^A$	Matriz que en la posición (i,j) contiene $\lambda^{a_{ij}}$.

Por ejemplo:

```
>> A=[1 5 7;2 6 9], B=2*ones(3)
>> C=A*B
>> C=A.*B % Da error
>> C=A.*A, C=B.^2, C=B^2, C=A.^2, C=2.^A, C=1./A
```

3.1.3. Funciones que actúan sobre matrices

Las siguientes funciones permiten obtener información sobre las matrices que tienen como argumentos de entrada:

Orden	Salida
<code>size(A)</code>	Vector con las dimensiones de la matriz A .
<code>size(A,1)</code>	Número de filas de la matriz A .
<code>size(A,2)</code>	Número de columnas de la matriz A .
<code>length(A)</code>	Mayor de las dimensiones de la matriz A . Equivale a <code>max(size(A))</code> . Si A es un vector, nos dice su longitud.
<code>diag(A)</code>	Vector cuyos elementos son los de la diagonal principal de A .
<code>rank(A)</code>	Rango de la matriz A .
<code>det(A)</code>	Determinante de la matriz A .
<code>inv(A)</code>	Inversa de la matriz A , cuando A es inversible.

Además, las funciones matemáticas habituales que el programa tiene predefinidas (trigonométricas, exponenciales, etc.) también pueden utilizarse sobre matrices o vectores, con la particularidad de que en ese caso actúan elemento a elemento. Así, por ejemplo:

Ejercicio 3.2

```
>> A=[0 pi;pi/2 3*pi/2];
>> cos(A) % matriz cuyos elementos son los cosenos de los elementos de A
>> B=[1+i,-1,1;2+2i,0,-5i];
>> C=abs(B) % matriz cuyos elementos son los módulos de los elementos de B
>> D=imag(B) % matriz cuyos elementos son las partes imaginarias de los de B
```

Ejercicio 3.3

1. Construir los vectores $\mathbf{v} = (1, 2, i)$ y $\mathbf{u} = \begin{pmatrix} 1 \\ 2 \\ i \end{pmatrix}$.
2. Construir el vector \mathbf{w} cuyas coordenadas sean los números impares comprendidos entre el 3 y el 25.
3. Almacenar en \mathbf{h} la coordenada novena de \mathbf{w} .
4. Con ayuda del apartado 2, calcular el vector $\mathbf{w1} = (2^3, 2^5, \dots, 2^{25})$
(Nótese que sus componentes no son equiespaciadas).
5. Construir un vector $\mathbf{w2}$, cuyas componentes formen una partición uniforme de 54 puntos del intervalo $[2, 3]$.

Ejercicio 3.4

1. Construir la matriz $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 7 & 9 & 11 \\ 8 & 10 & 15 \end{pmatrix}$.
2. Extraer la tercera columna de A y almacenarla en la variable \mathbf{v} .
3. Extraer el elemento de A que ocupa el lugar $(1, 3)$ y memorizarlo en la variable $\mathbf{a13}$.
4. Construir la matriz B que resulta de suprimir la segunda fila de A .
5. Construir la matriz C , resultante de cambiar la segunda columna de B por el vector $\mathbf{w} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$.
6. Construir la matriz D que resulta al añadir a la derecha de C el vector $\mathbf{b} = \begin{pmatrix} 7 \\ 2 \\ 7 \end{pmatrix}$.
7. Construir la matriz E que resulta de añadir a la matriz A una última fila con el vector $\mathbf{b} = (7, 2, 7)$.

Ejercicio 3.5

Dada la matriz $M = \begin{pmatrix} 2 & 2 & 2 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 1 & 0 \\ 2 & 2 & 2 & 0 & 0 & 1 \\ 0 & 0 & 3 & 3 & 3 & 3 \\ 0 & 0 & 3 & 3 & 3 & 3 \end{pmatrix}$

1. Almacenarla en la variable M , usando órdenes que eviten definirla elemento a elemento.
2. Calcular, caso de existir, su traspuesta y su inversa.

Ejercicio 3.6

Dadas las matrices:

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 7 & 9 & 11 \\ 8 & 10 & 15 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 5 \\ 2 & 8 \\ 1 & 8 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 2+i \\ 3i & 4 \\ 5 & 6 \\ 7 & 8 \end{pmatrix}$$

1. Calcular la matriz $D = A \cdot B$.
2. Calcular la matriz $D1$ que resulta de “pegar” a la derecha de D la matriz C .
3. Calcular el rango y el determinante de $D1$.
4. Calcular las matrices: C^t , \overline{C}^t , $D2 = A^t \cdot D1^{-1}$.
5. Calcular el número de filas y de columnas de D y de C .
6. Calcular la matriz $E = (e_{ij})$, tal que $e_{ij} = d_{ij}c_{ij}$, siendo d_{ij} y c_{ij} los elementos que ocupan los lugares (i, j) en las matrices D y C respectivamente.

Ejercicio 3.7

1. Calcular un vector \mathbf{v} cuyas coordenadas sean las raíces octavas de la unidad y comprobar que su suma es 0 y su producto -1.
2. Escribir en la ventana de comandos:

```
plot([v,1]), hold on, plot(v,'ro'), ezplot('x^2+y^2=1',[-1,1,-1,1])
```


y observar la figura resultante.

4. Determinantes

Ejercicio 4.1

Calcular el valor del siguiente determinante:

$$\begin{vmatrix} 1,3 & -2,3 & 3,2 & 4,3 \\ -2 & 4,3 & 6 & 8 \\ 3 & -6,7 & 9 & -13 \\ -4,2 & 8,9 & -3,3 & 1,7 \end{vmatrix}$$

Ejercicio 4.2 Dada la matriz:

$$\begin{pmatrix} 1+x & 1 & 1 & 1 \\ 1 & 1-x & 1 & 1 \\ 1 & 1 & 1+z & 1 \\ 1 & 1 & 1 & 1-z \end{pmatrix}$$

- Almacenarla en la variable M .
- Guardar en la variable N , sin utilizar la definición elemento a elemento, la matriz que resulta de substituir en M la x por 5 y la z por -17 (utilizar el comando **subs**).
- Resolver la ecuación: $|M| = 0$ (emplear el comando **solve**).

5. Sistemas de ecuaciones

Órdenes y funciones que pueden resultar de utilidad:

- **rref(A)**. Calcula la matriz escalonada reducida de la matriz A que proporciona el método de Gauss-Jordan.
- **solve(ec1,ec2,...,ecm,x1,x2,...,xn)**. Calcula las soluciones del sistema lineal de m ecuaciones $ec1=0$, $ec2=0$, ..., $ecm=0$, con n incógnitas: $x1$, $x2$, ..., xn .

Las incógnitas $x1, x2, \dots, xn$ son variables simbólicas. Las soluciones son calculadas en **orden lexicográfico** (por ejemplo: $a, b1, b2, c, x, y1, y2, z, \dots$).

Es aconsejable almacenar en dos variables, digamos A y b , la matriz de coeficientes y el vector columna de términos independientes. Así tendremos ya almacenados todos los datos de nuestro sistema. A partir de ellas podemos generar de forma sencilla la matriz ampliada del sistema. Resulta muy recomendable calcular el rango de la matriz del sistema y de la matriz ampliada para clasificar el sistema.

Nótese que aplicando el comando **rref** a la matriz ampliada, realmente estamos resolviendo el sistema mediante el método de Gauss-Jordan.

También podemos definir en otra variable, por ejemplo X , el vector columna que contenga las incógnitas del sistema. Así, podemos construir una matriz columna, $Ec=A*X-b$, con las m ecuaciones del sistema en forma homogénea. Aplicando directamente la orden **solve** a la variable Ec nos ahorraremos el trabajo de escribir, de una en una, todas las ecuaciones de nuestro sistema.

Ejercicio 5.1

$$\text{Resolver el sistema: } \begin{cases} a + b + c = 2 \\ a - b - c = 0 \\ a + b - c = 4 \end{cases}$$

```
>> syms a b c
>> ec1=a+b+c-2, ec2=a-b-c, ec3=a+b-c-4
>> [a1,b1,c1]=solve(ec1,ec2,ec3)
```

Al no especificar las incógnitas, MATLAB ha calculado las soluciones para a , b y c , en ese orden, que es el orden lexicográfico. Para finalizar, hacemos:

Otra opción para almacenar la solución es:

```
>> s=solve(ec1,ec2,ec3)
>> s.a
>> s.b
>> s.c
```

Si seguimos las recomendaciones comentadas anteriormente, podemos hacer:

```
>> syms a b c
>> A=[1,1,1;1,-1,-1;1,1,-1]; t=[2;0;4];
>> Amp=[A,t];
>> [rank(A),rank(Amp)]
>> X=[a;b;c]
>> Ec=A*X-t
>> [a1,b1,c1]=solve(Ec)
```

Si el número de incógnitas que intervienen en las ecuaciones es mayor que el número de ecuaciones debemos elegir qué incógnitas queremos que aparezcan como principales y las demás serán consideradas como parámetros. En otro caso, el programa elige como incógnitas las más próximas a x en el orden lexicográfico.

Ejercicio 5.2

Hallar u y v , de forma que se verifiquen las ecuaciones:
$$\begin{cases} u + v = 2x \\ u - v = 2y \end{cases}$$

```
>> clear, syms u v x y
>> f1=u+v-2*x, f2=u-v-2*y
>> [u1 v1]=solve(f1,f2,u,v)
>> [x1 y1]=solve(f1,f2)
```

Ejercicio 5.3

Dado el sistema:

$$\begin{cases} 1,1x + 2,21y + \frac{7}{5}z = -3,32 \\ -1,2x + 3,12y + 4,14z - t = -6,44 \\ -2,1y + 3z + 4t = 0,2 \\ -x + 1,6y + 2z + \frac{1}{2}t = -4,7 \end{cases}$$

1. Almacenar en las variables A , b y Amp , la matriz del sistema, el vector de términos independientes y la matriz ampliada, respectivamente.
2. Aplicar el teorema de Rouché-Frobenius para estudiar el carácter del sistema.
3. Obtener la reducida de Gauss-Jordan y verificar que la información que proporciona corrobora lo obtenido en el apartado anterior. Con dicha matriz, obtener manualmente sus posibles soluciones.
4. Caso de que el sistema sea compatible, emplear la orden **solve** para obtener sus soluciones y compararlas con las del apartado anterior.

Ejercicio 5.4

Dado el sistema:

$$\begin{cases} 2x - 4y & = -10 \\ x - 3y & + t = -4 \\ x & - z + 2t = 4 \\ x - 7y + 2z - 3t & = -11 \end{cases}$$

1. Emplear la orden `solve` para estudiar sus posibles soluciones.
2. Aplicar el teorema de Rouché-Frobenius para estudiar el carácter del sistema.
3. Analizar las consecuencias que se deducen de los resultados anteriores.

Ejercicio 5.5

Se considera el sistema de ecuaciones dependiente del parámetro real m :

$$\begin{cases} 3x + 3y - z = 0 \\ -4x - 2y + 2mz = 0 \\ 2x + 4y + 6z = 0 \end{cases}$$

Clasificarlo y resolverlo, en los casos en los que tenga solución, siguiendo los siguientes pasos:

- Hallar el valor del determinante de la matriz del sistema.
- Determinar el rango de la matriz de coeficientes mediante el comando `rank`.
- Analizar los dos resultados anteriores y razonar los distintos casos que aparecen.
- Resolver el sistema en cada caso que tenga solución.

Ejercicio 5.6

Estudiar el sistema:

$$\begin{cases} (\alpha + 3)x - y + z = 0 \\ 5x + (\alpha - 3)y + z = 0 \\ 6x - 6y + (\alpha + 4)z = 0 \end{cases}$$

según los valores del parámetro real α . Resolverlo cuando sea compatible indeterminado.

Ejercicio 5.7

Dada la matriz:

$$A = \begin{pmatrix} 2 & -4 & 1 \\ 1 & -3 & -2 \\ 3 & -4 & 3 \end{pmatrix}$$

comprobar que es regular y determinar su inversa, mediante el método de Gauss-Jordan, haciendo uso del comando `rref`.