Práctica 1. Introducción a Matlab.

Todos los scripts y funciones se copiarán en un Word de resultados, atendiendo a la Sección y número del ejercicio en el que nos encontremos.

Para aquellos scripts o funciones que sean llamados desde consola de comandos se copiará, además, la llamada realizada, así como el resultado generado, tanto numérico como figuras.

MATLAB diferencia entre las mayúsculas y minúsculas. No es lo mismo escribir EJEMPLO que Ejemplo o ejemplo.

SECCIÓN 1. Aprendiendo comandos, tipos de variables, funciones y operadores en Matlab.

Las siguientes órdenes pueden serte útiles para el desarrollo de esta práctica:

- ***help***: Te ofrece ayuda sobre una función, incluyendo algún ejemplo sobre su uso y otras funciones relacionadas. Por ejemplo, teclea *>>help clc*
- ***lookfor***: Lista todas las funciones en cuya ayuda aparece una palabra clave. Por ejemplo, si quisiéramos saber qué funciones tiene Matlab para manipular directorios podríamos teclear *>>lookfor directory* y a continuación teclear *>>help pwd*, *>>help dir*, etc. para saber cómo funciona cualquier función que nos interese.

En la siguiente tabla se indican algunas órdenes útiles en MATLAB.

ORDEN	***SIGNIFICADO DE LA ORDEN***
HELP	• **Ayuda sobre órdenes y funciones internas de Matlab.** • **Ayuda sobre nuestras propias funciones .m**
WHAT	**Da una lista de funciones .m en el directorio especificado (*Por ejemplo: >> what a:\ejemplos*)**
CD	• **Cambia al directorio padre o directorio anterior *(>> cd ..)*** • **Cambia al directorio que se especifique *(>> cd a:\ejemplos*)**
DIR	**Da una lista del contenido del directorio en el que me encuentro**
PWD	***¿Qué hace este comando?***

1. Haciendo uso del comando **HELP** puedes saber las funciones de los comandos **diary, who, clear, hold, format y edit.** ¿Qué diferencia existe entre los comando **edit** y **type?**

Las siguientes órdenes son interesantes para poder conocer los operadores definidos en Matlab. Algunos

2. Haz lo siguiente usando las líneas de comando de Matlab (indicadas por >>)

Operación	***Ejemplo***	***Indica los resultados de las operaciones***
Suma +	>>format short >>a=7 >>b=2; %variable escalar **>>suma=a+b; %**El ; hace que no se vea el resultado pero sí se ejecuta la orden o sentencia. **>>suma**	Resultados de a, b y suma:
Resta -	>>resta=a-b; **>>resta**	Resultado resta:
Multiplicación *	**>>multiplicar=a*b**	Resultado multiplicar:
División /	**>>div1=a/b**	Resultado div1:
***División ***	**>>div2=a\b** **>>div3=b\a %¿Es igual que div1?**	Resultado div2: Resultado div3:
Potencia ^	**>>a=3;potencia=a^2**	Resultado potencia:
	>>a=9; **>>a^3**	Resultado: *¿Cómo se llama la variable a la que se le asigna el resultado?*
	>>ans **>>c=7;d=8;ans=1** **>>(c*d)/(ans+7)** **>>(c*d)/ans+7**	*¿Cuáles son los valores que va tomando la variable por defecto ans?*

(a) Ejecuta en la línea de comandos el comando **who**. ¿Qué aparece en pantalla?
(b) Ejecuta: **>> clear b.** Pregunta por el valor de **b**. ¿Qué sale en pantalla?
(c) Ejecuta el comando **who**. ¿Cuál es la diferencia que encuentras con la ejecución anterior?

3 (Ejercicio para hacer en casa). Busca el significado de las siguientes **funciones internas** de

ABS		
SQRT		
RAND		
SIN		
SIND		
TAN		
ASIN		
SINH		
EXP		
LOG		
LOG10		
REM		
ROUND		
EPS		
PI		

4. Introducción a vectores. Como en todos los lenguajes de alto nivel las variables numéricas se clasifican básicamente en variables escalares (un número), variables vectoriales (vector) o variables matriciales. En ejemplos anteriores se ha visto la asignación numérica escalar, ahora vamos a ver una de las formas para asignar valores/elementos a un vector. Busca el significado de las órdenes ***LINSPACE*** usando el comando *help*, y ejecuta

>> x=linspace(0,2*pi,30)

Nota. Esta línea de comandos permite asignar valores numéricos a la variable llamada *x*. A diferencia de las asignaciones anteriores la variable x es un vector. Sin embargo, hay que aclarar

(d) Para determinar el número de elementos de un vector usa la función interna **length**. Indica la orden que has ejecutado para saber el número de elementos de los vectores x e y.

(e) La siguiente ejecución es incorrecta: **>> z=x^2.** Modifícala para que tengamos el cuadrado de cada uno de los elementos de vector x. Recuerda el ejercicio 1.

SECCIÓN 2. Variables

1. Indica los resultados tras ejecutar las siguientes órdenes. Si hubiera errores, corrígelos. Fíjate en las órdenes que han dado lugar a matrices o variables matriciales

```
>> X=[ 1 2  9 4  5]
>> X=[1,2,9,4,5]
>> X=[ 1;2;9;4;5]
>>X'
>>X(3)
>>X(0)
>>X([1,3,5])
>>X(1:4)
>>X=20:-2:1
>>X(end)
>>longitud=length(X)
>>tamano =size(X)
>>[fil,col]=size(X)
>>a=[1:10,2:12,3:13]
>> a=[1 2 7
5 6 8
3 2 6]
>> b=[1 2 3]
>> c=[a;b]
>>d=[1,2,3;4,3,0;6,8,1]
>>e=[1 2 3;4 3 0;6 8 1]
>>f=e', g=b';
```

2. Crea un vector *x* de 4 componentes equi-espaciado entre los valores 6 y 7.

3. Suma 1 al tercer elemento del vector x.

4. Crea un vector *y* de las mismas dimensiones que *x* con los primeros números impares.

5. Crea un vector v que contenga la tabla de multiplicar del 3, cuyo primer elemento sea cero y el último sea 30.

9. Escribe un vector z que vaya del 0 al 1 con un espaciado de 0.1. Calcula un vector con los valores de la función interna *sin* aplicada a los elementos de z. Hazlo también para la función interna *exp*.

10. Crea un vector x con las fracciones 1, 1/2 , 1/3, 1/4, … 1/10.

11. Genera los primeros 10 números de la serie: 0, 1/2, 2/3, 3/4, …

12. Crea la serie $(-1)^n$ con n= 0, …, 10

13. Crea un vector con los números de la serie: $\frac{(-1)^n}{2n+1}$ desde n = 0, …, 100 . Usa la función interna *sum* para sumar todos los elementos del vector. Repetir el paso anterior con 10000 términos. ¿Cuál de los dos resultados está más próximo al valor de $\pi/4$? Nota: $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}$ se corresponde con la fórmula de Leibniz para el cálculo de π.

14. Crea una matriz *B* cuya primera fila contenga el vector *a* del ejercicio 8, cuya segunda fila sea un vector de ceros, la tercera un vector de unos y la cuarta el contenido del vector *b* del ejercicio 8. Ver *ones y zeros.*

15. Define un vector *k* cuyos elementos sean iguales a los dos primeros elementos de la diagonal de la matriz *B*. Utiliza la función interna *diag.*

16. Crea una submatriz *C* de tamaño 3x3 que sea el contenido de las tres primeras filas y columnas de la matriz *B*.

17. Sean dos matrices *A* y *B*:

$$A = [1\ 2;\ 4\ -1], B = [4\ 2;\ -6\ 3]$$

(a) Calcula los resultados de las dos operaciones suma y resta entre matrices: *C = A+B* y *D=A-B.*
(b) Calcula los determinantes de ambas matrices y sus inversas.
(c) Calcula el resultado de: E=A*B y F=B*A.
(d) Empleando la operación elemento a elemento obtén la matriz G que sale de la siguiente expresión.

$$g_{ij}=a_{ij}-b_{ij}\cdot a_{ij}^{2/3}$$

Si observas algo extraño averigua el porqué.
(e) Tiene sentido realizar las siguientes operaciones *A/B* y *A\B*? Razona tu respuesta

18. Resolución de sistemas de ecuaciones: Sea el siguiente sistema de ecuaciones de la forma A·x=b, resuélvelo mediante los operadores matriciales estudiados en clase

$$A = [1\ 3\ 6;\ 8\ 9\ 1;\ 3\ 10\ 9];\ b = [1;\ 3;\ 4]$$

19. Veamos ahora el indexado lógico de matrices y vectores:
Sea el vector `x=[-1 0 2 4 -2 3 1 4 7 0 -3 -1];`. Si queremos asignar a cero los elementos de x mayores de cero hacemos en la línea de comandos `x(x>0) = 0`

(e) Crear un nuevo vector que tenga 1's en las posiciones de los elementos de x que sean mayores que la media (usar la función interna *mean*) y 0's en las de los menores que la media. NOTA: primero haz >>y=x %para evitar modificaciones del vector x y, por tanto, también de su media.
(f) Asignar a cero los elementos pares (usar función interna *rem)*.
(g) Cambiar el signo de los valores de x que verifican que 2<=x(i) < 5.

SECCIÓN 3. Funciones y scripts.

1. Introducción a un script (leer en casa). Un script es un programa que construye el usuario de Matlab. Este programa se construye con el lenguaje de alto nivel de Matlab. Todas las asignaciones que se realicen en el script o/y en la ventana de comandos se comparten. Todo script tiene que tener un nombre y una extensión fija .m. Esta extensión indica que el script o fichero está escrito en el lenguaje de Matlab. El script se puede crear con un editor (p.e. *edit*) o con la pestaña *File* del menú superior del entorno de matlab. Cuando se haya terminado la edición se graba (*save as*) con nombre y extensión .m (la extensión normalmente se pone por defecto, es decir no hay que ponerla). Tras la grabación el programa estará en el directorio donde se ha grabado (en nuestro caso en d:\alumnos\work). Al ejecutarlo en la ventana de comandos solo se escribe el nombre del script. Fíjate en los detalles. Los apóstrofos están en la tecla del cierre de interrogación.

Ejemplo de script	**Ejecución de un script en la ventana de comandos**
% Script ejemplo_script.m disp('Cuánto vale A_script y B_vc en ejemplo_script') A_script=37.5 disp([B_vc]) disp('Abandono el script')	>>edit ejemplo_script.m >>B_vc=5555; >> ejemplo_script %ejecución de un script >>disp('Cuánto vale a_script en ventana de comandos') >>A_script

IMPORTANTE: Matlab diferencia entre las letras mayúsculas y minúsculas como ya se indicó. Luego, en el ejemplo anterior si se pregunta por **A_script** dará el valor asignado 37.5, mientras que si se pregunta por **a_script** nos dirá que esta variable no existe.

IMPORTANTE: Como en todo lenguaje de alto nivel hay palabras reservadas, p.e. **disp**., que no

directorio en donde trabajabas teclea >>cd nombredeldirectorio. Si quieres trabajar con tu pen usando el comando cd indica a continuación el nombre con el que se identifica tu pen.

2. Introducción básica a funciones (leer en casa). Una función al igual que un script la tiene que crear un programador o usuario de Matlab. En Matlab hay dos tipos de funciones, función interna, como p.e. la función ya conocida ***sin,*** y función externa o creada por el usuario de Matlab. Centrándonos en esta última, una función tiene que tener un nombre y la extensión .m, se crea con un editor y se graba en un directorio. **CUIDADO**: La función se debe grabar con el mismo nombre que está indicado en la primera línea (la línea donde está la palabra reservada *function*). Se ejecuta en la ventana de comandos. Otro ejemplo de palabra reservada es **function**. Vas a realizar muchas funciones a lo largo del curso; a continuación sólo se muestra un ejemplo que debes hacer. Presta atención.

Ejemplo de función	**Ejecución de una función en la ventana de comandos**

function [c_fun,d_fun]=ejemplo_fun(a,b)
%ejemplo_fun.m
 	
disp('Cuánto vale a,b, c_fun y c_fun dentro de ejemplo_fun')
disp(a)
disp(b)
 	
c_fun=a+b;
d_fun=a*b;
 	
disp([c_fun d_fun]) % Fíjate en este disp.
disp('Abandono la función')
	>>a=10; b=5;
	>> [C_fun,D_fun]=ejemplo_fun(a,b)
	>> disp('Los valores numéricos de las variables c_fun y d_fun definidas dentro de la función se han asignado a las variables C_fun y D_fun en la ventana de comandos tras ejecutar la función ejemplo_fun en dicha ventana')
	>> disp([C_fun D_fun])
	>>% las variables a y b son las denominadas **variables o parámetros de entrada** a la función.
	>>% las variables C_fun y D_fun son las denominadas **variables o parámetros de salida,** obtenidos tras la ejecución de la función.
	>> disp('Sin embargo, en la ventana de comandos no conoce las variable c_fun y d_fun, ya que únicamente están definidas dentro de la función. La siguiente ejecución así lo dirá')

Script	***Líneas de ejecución del script en la ventana de comandos de Matlab***
	>> >>

4. Escribe una función que tenga como variables de entrada el radio y la altura, y como variables de salida el volumen y la superficie. Llama a esta función sup_vol_fun.m. Escribe la función y las órdenes.

Escribe la función	***Líneas de ejecución de la función en la ventana de comandos de Matlab***
	>> >>

5. Crea la función *suma_vectores* que tome como parámetros de entrada los vectores *a* y *b*, devolviendo un vector *c* que realice su suma. Hazlo también con un script, llámalo *suma_vectores_s.* ¿Qué diferencia aprecias entre *suma_vectores* y *suma_vectores_s*? ¿Las variables tienen el mismo ámbito? ¿La forma de invocarlos es la misma?

6. Escribe una función que reciba como argumento un vector *x* y un entero *n* y devuelva el valor de la función matemática: $f(x) = \frac{1}{1+|x|^n}$. Construye una figura que muestre cómo varía f(x) si n=3 y x varía entre -40 y 40. Pon etiquetas a los ejes usando los comandos xlabel e ylabel.

7. Crea una función llamada *alcance.m* que determine el alcance máximo horizontal (*xmax*), de un objeto lanzado con una velocidad inicial *vo* en el plano xy. Considera como variables de entrada el ángulo ϑ y la velocidad de origen *vo* del lanzamiento del objeto, ambas serán variables escalares. La variable de salida será el alcance máximo.

$$x_{max} = \frac{v_0^2 \cdot sen(2 \cdot \theta)}{g}$$

8. Haz lo mismo con un script, llámale alcance_s.m

9. Crea una función tiempo.m que determine el tiempo máximo del lanzamiento. Variables de entrada: ángulo y velocidad inicial. Variable de salida: tiempo máximo.

$$t_{max} = \frac{2 \cdot v_0 \cdot sen(\theta)}{g}$$

puede ser opcional. Si se omite debe tomarse como n=2 (devolviendo el cuadrado de x). Ver la función interna nargin. Usa la estructura if.

2. Crea una función *sumainvimp.m* que determine la suma de los inversos de los impares menores de 1000. Usa las funciones *tic* y *toc* para determinar el tiempo que se tarda. Usa la estructura for.

$$Serie = 1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \ldots + \frac{1}{999}$$

3. Los números de Fibonacci, F(n) son una serie de números enteros, inicializados con F(1)=0, y F(2)=1. Los demás términos se tienen como la suma de los dos anteriores: F(n) = F(n-1) + F(n-2). Escribe un script que genere los primeros 100 números de la serie de Fibonacci, almacenándolos en un vector F. Usa la estructura while.

4. Escribe las líneas de código para que dado un vector x, genere otro vector con el orden de los elementos invertido.

5. Implementa una función que tome como entrada un vector *x* y que devuelva la posición y el contenido del primer elemento negativo. Sintaxis de llamada: [elem, pos] = buscaelem(x). (a) Añadir una condición de parada si no hubiese ningún elemento negativo, que muestre por pantalla el mensaje: *No hay ningún elemento negativo.*

6. Implementa una función que dado un vector devuelva el menor valor del vector y la posición donde se encuentra sin usar la función *min* o *max*. Por ejemplo v=[1 2 3 4 5 6 -1 -3 -2] devolvería el valor de -3 y la posición 8.

7. Crea una función que dados los valores escalares de n, a_1, a_2 devuelva un vector de n componentes tal que $a_n = a_{n-1} - 3a_{n-2}$.

8. Crea un script que obtenga el valor medio y desviación media del vector *x* descrito como:

$$med = \frac{\sum_{i=1}^{n} x_i}{n}, desv = \sqrt{\frac{\sum_{i=1}^{n} (x_i - med)^2}{n}}$$

Compara los resultados con las funciones internas mean y std. Compara ahora el resultado de std con tu variable desv si la divides entre (n-1) en vez de n.

9. Implementa una función *SumoDiag.m* que tome como entrada una matriz cuadrada *A* y que devuelva la suma de todos los elementos de las 2 diagonales de dicha matriz. No utilizar la función *diag*().

10. Implementa una función *MultiplicoMatrices.m* que dadas dos matrices A y B definidas como $(a_{ij})_{mxn}$ y $(b_{ij})_{nxp}$, implemente la multiplicación de matrices C=A·B mediante bucles, definida como $(c_{ij})_{mxp}$ donde $c_{ij} = \sum_{n}^{r=1} a_{ir} \cdot b_{rj}$.

11. Crea una función llamada *alcance_bucle.m* que determine el alcance máximo horizontal y el

Ejemplo:

13. Crea la función *bin2dec_entero.m* que realice el funcionamiento inverso, dado un vector en

```
>>
dec2bin_entero(19)
ans =

  1   0   0   1   1
```

binario devuelva su valor en decimal.

```
>> bin2dec_entero([1 0
0 1 1])
ans =
```

14. Crea la función *dec2bin_decimal.m* que ante la entrada de un número fraccionario (sin parte entera, es decir, decimal positivo y menor que cero) y devuelva un vector con el contenido de los números en binario. Ejemplo:

```
>>
dec2bin_decimal(0.125
)
ans =
```

15. Crea una función, y el script correspondiente para llamarla, que reciba como parámetro de entrada un número entero en base 10, un tamaño de registro N y una opción. Según la opción dada deberá devolver:

- Opción = 1: la función devolverá el número binario en magnitud y signo, representado en un vector,
- Opción = 2: la función devolverá el número binario en complemento a dos, representado en un vector,
- Opción = 3: la función devolverá el número binario en exceso según el tamaño de registro, representado en un vector,

Hay que tener en cuenta que si el número no puede representarse por el tamaño del registro hay que notificarlo.

Crea un script que llame a la función con diferentes ejemplos, uno por opción, y recoja el resultado.

Ejemplos:

1. Número en decimal = -3, opción = 1 y N = 4, binario de salida [1011].
2. Número en decimal = -3, opción = 2 y N = 4, binario de salida [1101].

- Un entero negativo $-x$, donde $1 \leq x \leq 2^{(n-1)}$, se almacena como la representación binaria del entero positivo: $2^n - x$

El número así construido, recibe el nombre de complemento a 2 de x.
En general, dado un número entero $N < 2^n$ su complemento a dos en una representación binaria de n dígitos se define como: $C_2^N = 2^n - N$

Errores

16. Un algoritmo es inestable cuando los errores de redondeo se acumulan degradando la exactitud del resultado final, a pesar de que en aritmética exacta el algoritmo sea correcto. La sucesión siguiente es un ejemplo de algoritmo numérico inestable,

$$x_{n+2} = \frac{13}{3}x_{n+1} - \frac{4}{3}x_n, tomando\ como\ valores\ iniciales x_0 = 1 y x_1 = \frac{1}{3}$$

Se puede comprobar que el término general de esta sucesión es igual a:

$$x'_n = \left(\frac{1}{3}\right)^n = \frac{1}{3}x'_{n-1}$$

Si suponemos que x'_n son los valores exactos de la sucesión se puede estimar el error de redondeo de x_n. Escribir un programa en MATLAB ***(.m)*** que resuelva esta sucesión y visualice en pantalla, desde n igual a 1 hasta 20 iteraciones, x_n, x'_n, el error absoluto $|x_n - x'_n|$ y el error relativo $\left|\frac{x_n - x'_n}{x'_n}\right|$. Escribe el programa y los resultados que obtienes.

17. Matlab utilizan el formato IEEE754 en doble precisión para la representación de números reales:

a. Teniendo en cuenta que la mantisa es de 52 bits, corroborar que la precisión del computador es eps=2^{-52}
b. Realiza la siguiente operación a=1+2^{-52}, b=1-a. ¿Qué valor se obtiene?¿Por qué?
c. Realiza la siguiente operación a=1+2^{-53}, b=1-a. ¿Qué valor se obtiene?¿Por qué? ¿Qué tipo de error se produce y por qué?
d. Ejecuta el siguiente script. ¿Son correctos los resultados? Razona tu respuesta

```
k=1
while ((1.0+2^(-k))>1.0)
   k=k+1;
end
disp(['El número de bits de la matisa es de ',num2str(k-1)]);
```

de la función calcula_phi.m. Descarga del campus virtual calcula_phi.m y dibuja_valores.m. Ejecuta la *calcula_phi(0.01*)

a. Añade un punto de ruptura (*breakpoint*) en la línea 13.
b. Ejecuta una introducción paso a paso (*Step*). ¿Cómo cambia el valor de phi y por qué?
c. ¿Cómo va evolucionando el valor de E y por qué?
d. Coloca un *breakpoint* en la línea 16: "dibuja_valores(iter,phi,'b*')"
   i. ¿Qué diferencia observas entre *Step*, *Step In* y *Step Out*?
   ii. ¿Y con *Continue*?

SECCIÓN 6. Representación gráfica

1. Escribe la función *dibujo_parabola.m* que muestre paso a paso en una figura la trayectoria del tiro parabólico para v_o 40 m/s y θ=45º y para v_o igual a 60 m/s y θ=60º. En cada instante se mostrará con un círculo rojo la posición (x,y). Llama a alguna de las funciones anteriores. Usa la función *plot y hold on*

$$x = v_0 \cdot cos(\theta) \cdot t \qquad y = v_0 \cdot sen(\theta) \cdot t - \frac{1}{2} g \cdot t^2$$

(a) Añade el nombre de cada eje con *xlabel*, *ylabel* y el título de la figura con *title*
(b) Emplea los comandos de la ventana que contiene el gráfico para cambiar el grosor y el color de las líneas de las trayectorias.

2. Usa únicamente una sola figura para representar en diferentes paneles (usa subplot) las siguientes funciones. En el panel 1, dibujar la función $y = e^{(-x^2)}$en el intervalo [-2 2] empleando 100 datos equiespaciados y en los paneles 3 y 5, respectivamente, las funciones $y = e^{\left(-\left(\frac{x}{2}\right)^2\right)}$ y $y = e^{(-(2x)^2)}$. Cada función debe ir en un color y en un estilo de línea diferente.

Añade sólo en la figura del panel 5 el comando grid on. En el panel 6 se representará la última función en escala logarítmica en uno de los ejes. En el panel 2 se representará la primera función empleando *stem*. En el panel 4 se representará la segunda función empleando *hist*. ¿Tiene sentido lo que observas?

3. Crear una función *dibujatriangulo.m* que tenga como argumento de entrada los 3 vértices de un triángulo y devuelva una figura con el triángulo dibujado.

4. Crea una función *dibujaMUC.m* que dibuje la trayectoria circular de una partícula conociendo su velocidad angular *ω* y el radio de giro *r*. Recuerda el significado del periodo de un movimiento circular.
Las ecuaciones cartesianas son:

$$x = r \cdot cos(\omega \cdot t) \qquad y = r \cdot sen(\omega \cdot t)$$

(a) Añadir a la trayectoria en cada punto el vector velocidad con el comando *quiver*,

6. Crea una función *dibujatiroparabolico.m* que represente el tiro parabólico en 3D sabiendo que la función tiene como entrada los ángulos ϕ y θ de acuerdo a las siguientes expresiones:

$$x = v_0 \cdot cos(\theta) \cdot cos(\varphi) \cdot t \qquad y = v_0 \cdot cos(\theta) \cdot sen(\varphi) \cdot t \qquad z = v_0 \cdot sen(\theta) \cdot t - \frac{1}{2} g \cdot t^2$$

A la hora de representar la función, emplea el comando *plot3* y *pause* para ir viendo cómo se dibuja la trayectoria parabólica. Calcula el alcance máximo en cada eje previamente y fija de antemano los valores de los ejes empleando los comandos *xlim* ,*ylim* y *zlim* respectivamente.

7. Usando el Comando *meshgrid*, crea una retícula cuadrada en el intervalo x=[-1 1] y=[-2 2] emplea para ello un paso de malla de valor 0.1. Crea una matriz de ceros del tamaño de las matrices que definen la retícula. Representa, empleando el comando *mesh*, la matriz de ceros creada sobre la retícula. Representa gráficamente la superficie $z = e^{-(x^2+y^2)}$. Dibuja las curvas de nivel usando el comando *contour*

SECCIÓN 7. Entrada y salida

1. Usando el comando fprintf escribe en la pantalla el número pi con 10 decimales. Mira previamente las opciones de formato de fprintf.

2. Usando fprintf escribe el número e con 2 decimales en una línea y con 8 decimales en la siguiente.

3. Sea la variable x = 174, usa fprintf para ver en una sola línea en pantalla el valor de x como:

- un entero (%d),
- un entero reservando 4 columnas (%4d)
- un entero, reservando 4 columnas y rellenando con 0's (%04d)
- un entero con signo (%+d)
- un número real con 2 decimales (%.2f)
- un número real en notación científica (%e)

4. Usa el comando input para asignar el valor 222 a una variable. Pregunta a continuación por el valor de dicha variable en la línea de comando de matlab.

5. Crea un programa que, dado un número introducido por el usuario, devuelva por pantalla la tabla de multiplicar del número introducido.

6. Crea cinco vectores aleatorios denominados v1, v2, v3, v4 y v5, a continuación guarda su contenido en un fichero usando el comando *save* de las tres siguientes formas:

(a) Los vectores v deben tener la misma dimensión, ejecuta >> save ale1.txt v* –ascii, donde

utilizarlo. Ejecuta las siguientes órdenes >>save ale2 v* %no lo graba en formato legible ya que no usas la opción –ascii, el fichero sin extensión se llamará ale2 y v* indica, como sabes,que se grabaran en este fichero las variables que comienzan con v, >>clear v* % para borrar todas las variables que empiezan por v del workspace, >> who %para ver si existen variables que comienzan con v, >> dir ale* % para ver los ficheros que comienzan con ale, ¿qué ves?, >>load ale %recupera información junto con asignación, >>v1 %verás que recupera la información asignada a v1.

(c) Volvemos a generar cinco vectores de longitud diferente y grabamos su información legible en un fichero ale3.txt con save. Si usas load no podrás recuperar la información. Vete a la pestaña de File y activa la opción Import Data hasta terminar. Mira si existe alguna variable con el nombre ale3 en el workspace y visualiza su contenido, ¿qué ves?, ¿podrías utilizar la información que contiene?

SECCIÓN 8. Ejercicio aplicado

1. Lee el fichero de precipitaciones ***precip.dat*** usando el comando type. Observarás que contiene 14 columnas, la primera se corresponde con el campo "Año", de la segunda a la decimotercera que contiene los datos de precipitación de cada mes (Enero, Febrero.... Noviembre, Diciembre), la última columna contiene el campo "Precipitación Total Anual". Las unidades de precipitación vienen dadas en mm/mes o mm/año, según el caso.

En el fichero precip.txt hay dos tipos de errores:

(a) La precipitación de uno de los meses es muy superior a la precipitación total que debería corresponder a la suma de la precipitación mensual de ese año. Esto ocurre en dos años.

(b) En otros cuatro años la suma total de la precipitación es diferente a la precipitación acumulada o total para ese año. A diferencia de lo que ocurría en el caso (a), la diferencia entre los valores de precipitación de los distintos meses no es muy grande.

El **objetivo** de este ejercicio es **identificar los meses en los que se da alguno de estos dos errores y sustituirlo por el valor correcto**. Para ello debes realizar las siguientes operaciones:

A. Extrae las precipitaciones en cada uno de los meses haciendo uso de load. Asigna la información de la precipitación total anual en una variable llamada *agno* y el resto de las precipitaciones a una variable matricial llamada *mes*.

B. Dibuja en una gráfica la evolución de la precipitación anual del mes de enero en los diferentes años.

C. Dibuja en una única gráfica (usa el comando *figure* para crear una nueva figura antes de empezar a dibujar) la evolución de la precipitación anual para cada uno de los meses, junto con el campo "Precipitación Total Anual".

D. Crea un script llamado *precip_bien.m* que realice las siguientes operaciones:

- Si el dato de "Precipitación Total Anual" no difiere mucho de la suma de las precipitaciones mensuales (error tipo (b)), considera que el dato de precipitación total es erróneo y sustituyelo por el valor de esta suma.

El script debe incorporar la visualización por pantalla de lo siguiente: (1) año donde se ha detectado un error, (2) suma total mensual para ese año, (3) dato de precipitación total anual de ese año y (4) si el error es del tipo (a), el mes en el que se encuentra dicho error.

Por último, el script debe crear un fichero, llamado *precip_bien.dat*, en el que se guarden los datos iniciales con los errores corregidos.

E. Crea otro fichero con el nombre *precip_modificado.dat* que contenga lo siguiente: (1) los datos corregidos (2) una última fila, llamada *media*, en la que se guarde el dato correspondiente a la media aritmética de la precipitación mensual de cada año y (3) una última columna que contenga la media aritmética mensual de todos los años. Incorpora en este script la representación gráfica, en dos figuras separadas, de los valores medios indicados.