Laboratorio de Computación Científica Curso 2014-15

***PRÁCTICA 1. Introducción a Matlab***

**Alumno: ……………………………………………………………………**

**Prácticas de Introducción/Programación MATLAB y OCTAVE**

**Aprendiendo comandos, 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/Octave 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** |

1. Haciendo uso del comando **HELP** puedes saber las funciones de los comandos diary, who, clear, hold, format y edit.

Las siguientes órdenes son interesantes para poder conocer los operadores definidos en Matlab. Algunos de ellos los vas a utilizar frecuentemente, sería útil que grabaras la información:

**(a) help \**

**(b) help arith.**

**(c) help slash.**

**(d) help relop.**

**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;****>>suma** | ¿Cuál es la diferencia entre la 2ª y 3ª líneas de comando?**La 2ª tiene un comentario.**Resultado suma: **9** |
| ***Resta -*** | >>resta=a-b;**>>resta** | Resultado resta: **5** |
| ***Multiplicación \**** | **>>multiplicar=a\*b** | Resultado multiplicar: **14** |
| ***División /*** | **>>div1=a/b** | Resultado div1: **3.5000** |
| ***División \*** | **>>div2=a\b****>>div3=b\a %¿Es igual que div1?** | Resultado div2: **0.2857**Resultado div3: **3.5000****Sí** |
| ***Potencia ^*** | **>>a=3;potencia=a^2** | Resultado potencia: **9** |
|  | **>>a=9;****>>a^3** | Resultado: **729**¿Cómo se llama la variable a la que se le asigna el resultado? **ans** |
|  | **>>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*? **ans1=7 ; ans2=15** |

1. Ejecuta en la línea de comandos el comando **who**. ¿Qué aparece en pantalla? **El nombre de tus variables.**
2. Ejecuta: **>> clear b.** Pregunta por el valor de **b**. ¿Qué sale en pantalla? **Que la variable “b” no está definida.**
3. Ejecuta el comando **who**. ¿Cuál es la diferencia que encuentras con la ejecución anterior? **No aparece la variable “b” porque ejecuté el comando >>clear b que la “elimina”.**

3. Busca el significado de las siguientes funciones internas de Matlab, pon un ejemplo y su resultado. Averigua cuál se puede escribir en minúsculas, mayúsculas o indistintamente. Normalmente en el segundo párrafo se especifica la sintaxis de la función. Las funciones trigonométricas deben tener el argumento en radianes.

|  |  |  |
| --- | --- | --- |
| ***Función*** | ***Ejemplo*** | ***Resultado ejemplo*** |
| ***ABS*** | ***>>abs(-5)*** | ***5*** |
| ***SQRT*** | ***>>sqrt(9)*** | ***3*** |
| ***RAND*** | ***>>rand(3,4)*** |  ***0.8147 0.9134 0.2785 0.9649*** ***0.9058 0.6324 0.5469 0.1576*** ***0.1270 0.0975 0.9575 0.9706*** |
| ***SIN*** | ***>>sin(pi)*** | ***1.2246e-016*** |
| ***TAN*** |  |  |
| ***ASIN*** |  |  |
| ***SINH*** |  |  |
| ***EXP*** |  |  |
| ***LOG*** |  |  |
| ***LOG10*** |  |  |
| ***REM*** |  |  |
| ***ROUND*** |  |  |
| ***EPS*** |  |  |
| ***PI*** |  |  |

**Aprendiendo a navegar en el entorno de Matlab/GUIoctave**

1. Crear un fichero M-file con el nombre *prueba.m* y escriba la siguiente línea disp('Hola Mundo')

¿Cómo se ejecuta el *script* creado?

¿Qué hace el script?

¿Cómo se puede introducir un comentario?

¿Cómo podemos saber cuál es nuestro directorio de trabajo?

¿Cómo se puede ir al directorio padre?

1. Defina el vector *x=[10:-1:1]* y la estructura *s=struct('a', x, 'b', 'Nombre', 'c', int16([x;x]))*. Visualice su contenido en el workspace.

¿Qué tipos de datos son *x* y *s*?

¿Cómo se puede cambiar el contenido del campo b de s?

**Evalúa tus conocimientos adquiridos con MATLAB/OCTAVE**

Estudiar cómo crear, concatenan e indexar vectores y matrices. Para evaluar tus conocimientos te recomendamos que escribas los comandos para realizar las siguientes operaciones

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

>>**x=linspace(6,7,4)**

>>**x =**

 **6.0000 6.3333 6.6667 7.0000**

* 1. Añadir 1 al tercer elemento del vector

**>> x=linspace(5,7,4);x(3)=1**

**x =**

 **5.0000 5.6667 1.0000 7.0000**

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

**>> x=1:1:10; y=1:2:20**

* 1. Crear una matriz *A* cuya primera fila corresponda al vector *x*, cuya segunda fila sea un vector de 0's, la tercera un vector de 1's y la cuarta el contenido de *y*
	2. Crear la matriz *B* cuyo contenido de la primera columna corresponda al vector *x*, y la segunda columna sea el contenido del vector *y*
	3. Definir un vector *z* cuyos elementos sean iguales a los 3 primeros elementos la diagonal de la matriz *A*. Utilizar el comando *diag()* y estudiar su funcionamiento mediante la función help (sintaxis *>>help diag*);
	4. Crear una submatriz *C* de tamaño 3x3 que sea el contenido de las tres primeras filas y columnas de la matriz *A*.
	5. Crear las matriz *D* y *E* con todos los elementos a cero y a uno.
	6. Obtenga los tamaños de la matriz *A* y *B*.
	7. Extraer la fila 2 de la matriz D y la columna 1 de la matriz E
	8. Estudiar los operadores aritméticos matriciales
	9. Crear dos matrices *A* y *B*:

Realizar las siguiente operaciones *C = A+B* y *D=A-B.* ¿Qué ocurre?

Realizar las siguiente operaciones E=A\*B y F=B\*A*.* ¿Qué ocurre?

Empleando la operación elemento a elemento obtén *G* cuyos elementos vienen descritos por

¿Son A y B singulares? Utilizar para ello el comando *det()* y *rcond()*. Estudiar la funcionalidad de ambas funciones. Calcular la inversa de las matrices si son singulares con el comando *inv()*.

Tiene sentido realizar las siguientes operaciones *A/B* y *A\B*? Razonar la respuesta

* 1. Resolución de sistemas de ecuaciones

Sea el siguiente sistema de ecuaciones de la forma A·x=b, resolverlo mediante los operadores matriciales estudiados en clase

**Funciones y scripts en Matlab/Octave**

1. Diseñar una función que calcule la superficie y volumen de un cilindro dados su radio *r* y altura *h.*

**>>function [S,V]=cilindro(r,h)**

**>>S=2\*pi\*r.\*h+2\*pi\*r.^2;**

**>>V=pi\*r.^2.\*h;**

1. Crear la función *suma\_vectores* que tome como parámetros de entrada los vectores *a* y *b*, devolviendo un vector *c* que realice su suma. Añadir una condición de error a la función anterior si las dimensiones de los vectores son distintas.
2. Crear un script denominado s*uma\_vectores\_script* que implemente la misma funcionalidad que s*uma\_vectores*.
3. ¿Qué diferencia aprecias entre *suma\_vectores* y *suma\_vectores\_script*? ¿Las variables tienen el mismo ámbito? ¿La forma de invocarlos es la misma? Razona tu respuesta.

**Bucles y condicionales**

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

**>>function x2=f(x)**

**>>n=length(x);**

**>>for i=1:n**

**>>x2(i)=x(n-i+1);**

**>>end**

1. Implementar una función que tome como entrada una matriz *A* y que devuelva la suma de todos los elementos de las 2 diagonales de dicha matriz. No utilizar la función *diag*().

**>>function sum=sumdiag(A);**

**>>[nf,nc]=size(A);**

**>>if nc~= nf**

 **>>error('no es una matriz cuadrada')**

**>>else**

 **>>sumdiag1=0;%inicializo**

 **>>sumdiag2=0;**

 **>>for i=1:nc**

 **>>for j=1:nc**

 **>>if i==j**

 **>>sumdiag1=sumdiag1+A(i,j);**

 **>>end**

 **>>if i+j==nc+1**

 **>>sumdiag2=sumdiag2+A(i,j);**

 **>>end**

 **>>end**

 **>>end**

 **>>sum=sumdiag1+sumdiag2;**

**>>end**

1. Implementar 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)
2. Añadir una condición de parada si no hubiese ningún elemento negativo, que muestre por pantalla el mensaje: *No hay ningún elemento negativo*
3. Añadir una condición de error si el vector de entrada está vacío y que muestre el mensaje de error por pantalla.
4. Implementar 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.
5. Implementar una función que dadas dos matrices A y B definidas como (aij)mxn y (bij)nxp, implemente la multiplicación de matrices C=A·B mediante bucles, definida como (cij)mxp donde .

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

1. Crear una función que dados los valores escalares de n, a1, a2 devuelva un vector de n componentes tal que an = an-1 - 3an-2.
2. Crear 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.

* 1. Construye una tabla que refleje este alcance máximo para diferentes valores del ángulo de inclinación variando desde 10º a 90º con un incremento de 5º y con una velocidad de origen de 60 m/s. NOTA: La expresión del alcance máximo viene depende también de la aceleración de la gravedad *g*.
	2. Haciendo uso de *alcance.m* construye otra función *alcance2.m* que además tenga como variable de salida el alcance máximo y el tiempo total que el objeto tarda en realizar el tiro parabólico. Añade a la tabla anterior los datos de esta nueva variable.

* 1. Dada la expresión de la trayectoria en el plano xy, construye la función *dibujo\_parabola.m* que muestre en una figura la trayectoria de ese tiro parabólico con vo=60 m/s y θ=45º empleando la función *plot*. Considera sólo como variable de entrada un vector *x* que contenga los diferentes alcances horizontales con los que deseas trabajar.

**Representación gráfica**

1. Representa la función en el intervalo [-2 2] empleando 100 datos equiespaciados.

* 1. Representa en el mismo gráfico, empleando el comando *hold on*, las siguientes versiones escaladas de la misma función: , . Utilizando un color distinto y un tipo de línea distinta para cada función.

* 1. Añade una trama al gráfico, mediante el comando *grid on*.
	2. Añade el nombre de cada eje con *xlabel*, *ylabel* y el título de la figura con *title*
	3. Emplea los comandos de la ventana que contiene el gráfico para cambiar el grosor de las líneas.
	4. Repite el proceso cambiando la escala del eje de forma que sea logarítmico.
	5. Vuelve a representar la función original, en otra ventana gráfica empleando la función *stem*.
	6. Emplea la función *hist* para representar los valores de y ¿Tiene sentido lo que observas?
1. Crear una función *dibujatriangulo* que tome los 3 vértices definidos como un vector (x,y) y dibuje el triángulo. Sintaxis: dibujatriangulo(a,b,c).
	1. La ecuación de una recta que pasa por dos puntos es:

1. Crear una función *dibujaMUC* que dibuje la trayectoria circular de una partícula conociendo su velocidad angular *ω* y el radio de giro *r*.
	1. Las ecuaciones cartesianas son:

* 1. Añadir el vector velocidad con el comando ***quiver*,** mediante las ecuaciones de la velocidad:

* 1. Crear una función *dibujatrayectoria3D* que tenga como parámetros de entrada el radio *r* y la velocidad angular *ω* sabiendo que la partícula se mueve en el eje z con una velocidad constate vz. Dibujar la trayectoria haciendo uso de la función *dibujaMUC* y del comando *plot3*.
1. Crear una función *dibujatipoparabolico* que represente el tiro parabólico en 3D sabiendo que la función tiene como entrada los ángulos φ y θ de acuerdo a las siguiente expresiones:

* 1. A la hora de representar la función, emplea el comando *plot3* y *pause* para ir viendo como 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.
1. **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.**
	1. **Crea una matriz de ceros del tamaño de las matrices que definen la retícula.**
	2. **Representa, empleando el comando *mesh*, la matriz de ceros creada sobre la retícula.**
	3. **Representa gráficamente la superficie**

* 1. **Dibuja las curvas de nivel usando el comando *contour***

**Entrada/salida**

1. Entrada y salida con ficheros
	1. Crear 5 vectores aleatorios denominados v1, v2, v3, v4 y v5 de tamaños diferentes y guardar su contenido en fichero usando el comando *save*.
		1. Recuperar la información usando *load* verificando que la información guardada y leída es la misma.
		2. Realizar la misma operación creando un fichero de texto plano y visualizarlo con un editor de textos
	2. Leer el fichero de precipitaciones anual precip.txt que contiene los campos (Año, Enero, Febrero, .... Noviembre, Diciembre, Total) y realizar las siguiente operaciones:
		1. Extraer las precipitaciones en cada uno de los meses
		2. Dibujar en una gráfica de la evolución de la precipitación anual del mes de enero en los diferentes años.
		3. Chequear si la columna total corresponde al total de las precipitaciones de los meses parciales, si no fuera así mostrar un mensaje de error por pantalla donde se avise de ello indicando el año en que no es coherente la información.
		4. Añadir la columna *media* que corresponde a la media aritmética de la precipitación mensual y guardarla en un fichero con el nombre de precip\_modificado.txt.
2. Entrada y salida por pantalla
	1. Crear un programa que dado un número introducido por el usuario, devuelva por pantalla la tabla de multiplicar del número introducido.
	2. Imprima por pantalla los campos de la estructura *s=struct('a', [1.5 7,3], 'b', 'Nombre', 'c', int16([x;x]))*, de acuerdo al tipo de dato de cada uno de los campos usando el comando *fprintf*.
	3. Cree un script que dado un número *n* introducido por pantalla (emplear para ello el comando *input*), guarde en un fichero la secuencia 1, fact(2)…., fact(n)

**Formato IEEE754 y errores**

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

* 1. **Teniendo en cuenta que la mantisa es de 52 bits, corroborar que la precisión del computador es eps=2-52**
	2. Realiza la siguiente operación a=1+2-52, b=1-a. ¿Qué valor se obtiene? ¿Por qué?
	3. 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?
	4. Ejecuta el siguiente script. ¿Son correctos los resultados? Razona tu respuesta

k=1;

while ((1.0+2^(-k))>1.0)

 k=k+1;

end

fprintf(1,'El número de bits de la mantisa es %d\n', k-1);

fprintf(1,'El eps calculado es %e\n', 2^-(k-1));

* 1. ¿Qué tipo de errores se producen cuando se teclea las siguiente cantidades 10^308 y 10^309?

**Números en binario**

1. Crea la función dec2bin\_entero que ante la entrada de un numero entero devuelva un vector con el contenido de los números en binario. NOTA: no utilizar la función bin2dec. Ej:
2. Crea la función bin2dec\_entero que realice el funcionamiento inverso, dado un vector en binario devuelva su valor en decimal.

>> dec2bin\_entero(19)

ans =

 1 0 0 1 1

>> bin2dec\_entero([1 0 0 1 1])

ans =

 19

1. **Crea la función dec2bin\_decimal que ante la entrada de un número fraccionario (sin parte entera) devuelva un vector con el contenido de los números en binario.**

>> dec2bin\_decimal(0.125)

ans =

 0 0 1

1. Crea la función dec\_to\_ieee754 que implemente la conversión al formato IEEE754 en simple **(y doble, añadido hoy)** precisión. La función tendrá la siguiente cabecera [ieee, signo, mantisa, ex]=dec\_ieee754(x) donde:

- ieee es el vector con el número codificado en dicho formato

- signo corresponde al bit de signo en dicho formato

- mantisa corresponde al vector mantisa en dicho formato

- exponente corresponde al campo exponente en dicho formato

>> [i,s,m,e] = dec\_to\_ieee754(-0.125)

i = 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

s = 1

m = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

e = 0 1 1 1 1 1 0 0

1. Crear la función análoga ieee754\_to\_dec que dado un número en formato IEEE754 en simple precisión devuelva el número convertido en decimal.

**Estabilidad de un algoritmo**

1. 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,

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

Si suponemos que *x'n* son los valores exactos de la sucesión se puede estimar el error de redondeo de *xn*. Escribir un programa en MATLAB ***(.m)*** que resuelva esta sucesión y visualice en pantalla, desde *n* igual a 1 hasta 20 iteraciones, *xn, x'n,* el error absoluto y el error relativo . Escribe el programa y los resultados que obtienes.

