UNIVERSIDAD DE CONCEPCION
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA MATEMATICA

Métodos Numéricos (525370)

Laboratorios Matlab 2024-2

Introducción al MATLAB – I

MATLAB (`www.mathworks.com`) es un lenguaje de programación y una herramienta de cálculo.

Un comando MATLAB puede terminar con ";" o no. Cuando se ejecuta un comando terminado en ";", los contenidos de las variables involucradas no se muestran en la pantalla. A continuación daremos una serie de comandos que muestran como trabajar con escalares, vectores y matrices.

```
>> a=1; % Un escalar, ingreselo con y sin ";".
```

Para ingresar el vector fila $v = (1\ 3\ 5\ 7)$:

```
>> v=[1 3 5 7]; % Las componentes van separadas una de otra por un espacio.
```

Para ingresar el vector columna $w = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$:

```
>> w=[1;3;5;7]; % Una fila va separada de otra por ";".
```

Muchas veces los vectores tiene una ley de formación. Esto permite una mayor facilidad para ingresarlos al computador. Por ejemplo, un vector con 100 componentes, donde la primera componente es 2, la última es 200 y las componentes intermedias van incrementados de dos en dos, puede ingresarse de la siguiente manera abreviada:

```
>> q=2:2:200; % El primer numero indica la componente inicial, el segundo el
>>            % incremento y el ultimo la componente final.
```

Cuando el incremento está ausente, se presupone el valor 1: así, son equivalentes

`r=1:45;` y `r=1:1:45;`

A continuación mostraremos ejemplos de algunas operaciones con vectores que pueden realizarse (hágalas una a una):

```
>> u=[1 2 3 4 5 6 7 8]; % Se ingresa un vector (fila).
>> v=8:-1:1 % Se ingresa otro vector (fila).
>> u+v  % Suma de vectores.
>> v'   % Vector transpuesto (columna).
>> u*v' % El vector u por el v transpuesto (producto interior entre dos vectores).
>> sqrt(u*u') % Norma del vector u ("sqrt" calcula la raiz cuadrada)
>> sin(u)     % Produce un vector de la misma longitud de u donde cada componente es
>>            % el seno de cada componente de u.
>> cos(u)     % Idem con coseno.
>> u.*v % Vector cuyas componentes son los productos de las componentes de u por
>>      % las de v (notar el . antes del signo *).
>> u./v % Vector cuyas componentes son las divisiones de las componentes de u por
>>      % las de v (notar el . antes del signo /).
>> u.^3 % Vector cuyas componentes son los cubos de las componentes de u
>>      % (notar el . antes del signo *).
>> 5^4  % Para elevar un escalar a una potencia no es necesario usar el punto.
>> length(v)  % Entrega el numero de componentes del vector v (longitud de v).
```

Para ingresar la matriz $M = \begin{pmatrix} 1 & 2 & 5 \\ 2 & -1 & 6 \\ 3 & 0 & -1 \end{pmatrix}$:

```
>> M=[1 2 5; 2 -1 6; 3 0 -1] % Una matriz se ingresa por filas. Los elementos
>>                           % de una misma fila se separan por un espacio y
>>                           % para separar una fila de otra se usa ";".
>> IM=inv(M)                 % Inversa de la matriz.
>> TM=M'                     % Transpuesta de la matriz.
>> Det=det(M)                % Determinante de la matriz.
>> VP=eig(M)                 % Valores propios de la matriz.
```

Notar que un escalar es una matriz 1×1 y un vector columna es una matriz $n \times 1$.

A continuación mostraremos ejemplos de algunas operaciones que pueden realizarse con matrices (hágalas una a una):

```
>> A=[1 2 5 5; 2 -1 6 0; 3 0 -1 4; -1 2 4 8; 1 2 3 6];  % Se ingresa la matriz A.
>> A(2,3)        % Muestra el elemento que esta en la posicion (2,3).
>> A(:,4)        % Muestra la cuarta columna de la matriz A.
>> A(2,:)        % Muestra la segunda fila de la matriz A.
>> A(1:3,2)      % Muestra desde el elemento 1 al 3 de la columna 2 de la matriz A.
>> [m,n]=size(A) % Muestra los numeros de filas (m) y columnas (n) de la matriz A.
```

Los siguientes comandos permiten construir matrices preestablecidas:

`eye`	Matriz identidad.
`zeros`	Matriz de ceros.
`ones`	Matriz de unos.
`diag`	Si x es un vector, diag(x) crea una matriz diagonal cuya diagonal son las componentes de x. Si A es una matriz cuadrada, diag(A) es un vector formado por la diagonal de A.

`triu`	Parte triangular superior de una matriz.
`tril`	Parte triangular inferior de una matriz.
`rand`	Matriz generada aleatoriamente con valores entre 0 y 1.
`hilb`	Matriz de Hilbert.

Los comandos anteriores combinados permiten ahorrar tiempo en la construcción de algunas matrices. Por ejemplo:

```
>> A=[1 2; 5 -2]
>> B=[-10 30; A]
>> C=[eye(2) zeros(2,2); zeros(2,2) A]
>> D=diag(diag(C))
```

MATLAB permite hacer gráficos, mediante el comando `plot`. Por ejemplo:

```
>> x=0:.01:10;
>> y=sin(x);
>> plot(x,y)
>> plot(x,y,'r')  % Note la diferencia con el anterior.
>> plot(x,y,'*')  % Note la diferencia con los anteriores.
>> plot(x,y,'*y') % Note la diferencia con los anteriores.
>> z=sin(x).^2;
>> plot(x,y,'r',x,z,'b') % Asi pueden dibujarse dos curvas en un mismo grafico.
```

También pueden hacerse varios gráficos a la vez agregando el comando `subplot`. Por ejemplo:

```
>> x=1:.01:10;
>> y=sin(4*x);
>> subplot(2,2,1) % Se divide la pantalla grafica en dos filas por dos columnas y
>>                % se utiliza la primera ventana.
>> plot(x,y)
>> subplot(2,2,2) % Estamos usando la segunda ventana.
>> plot(x,y,'r')  % Note la diferencia con el anterior.
>> subplot(2,2,3) % Estamos usando la tercera ventana.
>> plot(x,y,'*')  % Note la diferencia con los anteriores.
>> subplot(2,2,4) % Estamos usando la cuarta ventana.
>> plot(x,y,'*y') % Note la diferencia con los anteriores.
```

Para borrar los contenidos de todas las variables se usa el comando `clear`.

Para conocer la sintaxis correcta de alguna sentencia se usa el comado `help`. Por ejemplo:

```
>> help plot
```

Importante: MATLAB diferencia entre mayúsculas y minúsculas. Por lo tanto, "`a`" y "`A`" son variables diferentes!

UNIVERSIDAD DE CONCEPCION
FACULTAD DE CIENCIAS
FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA MATEMATICA

Métodos Numéricos (525370)

Laboratorio 2 2023-2

Introducción al MATLAB – II

En este laboratorio discutiremos los tipos de programas que pueden hacerse en MATLAB y cómo almacenar datos.

Hay dos tipos de programas MATLAB: uno se donomina *rutero* y el otro *function*.

Supongamos que tenemos un directorio donde guardaremos nuestros programas. MATLAB debe estar direccionado a ese directorio. Un comando para cambiar de directorio dentro de MATLAB es:

```
>> cd directorio
```

Todos los archivos con programas MATLAB deben terminar con la extensión `.m`. Veamos un ejemplo:

Deseamos resolver la ecuación de segundo grado $3x^2 + 5x + 2 = 0$. Escribamos primeramente un programa tipo *rutero*. El programa puede ser el siguiente:

```
a=3;
b=5;
c=2;
D=b^2-4*a*c;
x(1)=(-b+sqrt(D))/(2*a);
x(2)=(-b-sqrt(D))/(2*a);
x
```

Guarde el programa con el nombre `eje1.m`. Para ejecutarlo escriba en MATLAB el nombre del archivo y obtendrá:

```
>> eje1

x =

    -0.6667    -1.0000
```

Este tipo de programas se conocen como *ruteros* y las variables son globales, es decir, quedan en la memoria despues de ejecutarse el programa. Para saber que hay en la memoria puede usarse el comando `whos`:

```
>> whos
 Name      Size          Bytes  Class

  D         1x1              8  double array
  a         1x1              8  double array
  b         1x1              8  double array
  c         1x1              8  double array
  x         1x2             16  double array
```

Una desventaja de este tipo de programas es que para resolver otra ecuación que utilice la misma fórmula debemos modificar el programa.

Los programas tipo *function* tienen una estructura más esquematizada y siempre comienzan de la siguiente forma:

`function [`*salida1,salida2,...*`]=`*nombre*`(`*entrada1,entrada2,...*`)`

El programa anterior escrito como function queda asi:

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

Se almacena en un archivo `eje2.m` y se ejecuta del siguiente modo:

```
>> eje2(3,5,2)
ans =

      -0.6667   -1.0000
```

Este programa puede usarse, sin modificarlo, para resolver otras ecuaciones del mismo tipo. También puede usarse en otros programas (como veremos en otros laboratorios).

En este caso las variables son locales. Por ello si se ejecuta `whos` se obtiene:

```
>> whos
  Name      Size          Bytes  Class

  ans       1x2              16  double array
```

Es conveniente usar programas tipo *function*, cuando sea posible, pues permiten un ahorro de memoria.

A continuación daremos los comandos más usados en programas:

- `for`. La sintaxis de este comando es

```
for i=vi:in:vf
    instrucciones
end
```

  donde `vi`, `in` y `vf` son el valor inicial, el incremento y el valor final de la variable escalar i. Cuando `in` está ausente, se presupone el valor 1: así, son equivalentes `for i=vi:vf` y `for i=vi:1:vf`.

- `while`. La sintaxis de este comando es

```
while relación
    instrucciones
end
```

  Las instrucciones se ejecutan reiteradamente mientras la relación sea verdadera.

- `if`. La sintaxis de este comando es

```
if relación
    instrucciones
end
```

  Las instrucciones se ejecutan si la relación es verdadera. Otras formas de este comando son posibles. Por ejemplo,

```
if relación
    instrucciones 1
else
    instrucciones 2
end
```

  Si la relación es verdadera se ejecutan las instrucciones 1, caso contrario se ejecutan las instrucciones 2.

Las relaciones para los comandos `if` y `while` se construyen mediante los siguientes relacionadores:

`<`	menor que
`>`	mayor que
`<=`	menor o igual a
`>=`	mayor o igual a
`==`	igual a
`~=`	distinto a

y los siguientes conectivos lógicos:

`&`	y
`|`	o
`~`	no
`xor`	o excluyente

A continuación mostraremos un par de ejemplos de programas (prográmelos) y dejaremos algunos ejercicios (trate de hacerlos todos en el tiempo del laboratorio).

1. Construya un programa que evalúe la función $f(x) = \begin{cases} 2\,\mathrm{sen}^2(2x), & x \leq 0, \\ 1 - \mathrm{e}^{-x}, & x > 0. \end{cases}$

   Solución:

```
function y=fun1(x) % Si la entrada es un vector, la salida tambien lo es.
n=length(x);       % Determina la longitud del vector x.
                   % A continuacion se calcula el valor de la funcion
                   % componente a componente.
for i=1:n          % Al omitir el incremento este se asume igual a 1.
    if x(i)<=0
        y(i)=2*(sin(2*x(i)))^2;
    else
        y(i)=1-exp(-x(i));
    end
end
```

   Para hacer la gráfica de la función f en el intervalo $[-10, 10]$ puede utilizarse este programa del siguiente modo:

```
>> x=-10:.01:10;
>> plot(x,fun1(x))
```

2. Construya un programa que evalúe la función: $f(x) = \begin{cases} x - 1, & x \leq -2, \\ 1 - x^2, & -2 < x < 0, \\ -\dfrac{1}{x+1}, & x \geq 0. \end{cases}$

   Solución:

```
function y=fun2(x)
n=length(x);
for i=1:n
    if x(i)<=-2
        y(i)=x(i)-1;
    elseif (x(i)>-2 & x(i)<0)
        y(i)=1-x(i).^2;
    else
        y(i)=-1/(x(i)+1);
    end
end
```

   Utilice este programa para hacer gráficos de la función en diferentes intervalos.

3. Construya una función que genere una matriz de la forma $\boldsymbol{A} = \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}$.

Solución:

```
function A=matriz(n)
B=[zeros(n-1,1) eye(n-1);zeros(1,n)];
A=2*eye(n)-B-B';
```

genere y visualice la matriz $\boldsymbol{A}$ para $n = 8$. Genere el vector $\boldsymbol{b} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^8$ y resuelva el sistema $\boldsymbol{Ax} = \boldsymbol{b}$ mediante el comando MATLAB:

```
>> x=A\b
```

Verifique que el vector $\boldsymbol{x}$ obtenido resuelve el sistema de ecuaciones anterior evaluando (obviamente en MATLAB) $\boldsymbol{Ax} - \boldsymbol{b}$. ¿Qué observa?

4. Construya una función que evalúe e^x mediante su serie de Taylor: $\mathrm{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

   Solución:

```
function y=miexp(x)
y=1;
sum=x;
n=1;
while (y+sum~=y)
   y=y+sum;
   n=n+1;
   sum=x*sum/n;
end
```

   Explique por qué este programa siempre se detiene.

   Compare los valores de esta función con los de la función MATLAB `exp(x)` para distintos valores de x. Para visualizar más dígitos decimales utilice el comando:

```
>> format long
```

5. Usando el comando `help` de MATLAB estudie la sintaxis de los comandos `save` y `load`. Pruébelos con algunos de los programas realizados.

6. Dada una matriz $\boldsymbol{A}$, indique qué calculan las siguiente líneas MATLAB:

```
>> max(sum(abs(A')))
>> norm(A,inf)
```

FST