Universidad de Concepción
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

Cálculo Numérico (521230)
Laboratorio 1: Introducción al MATLAB

MATLAB es una abreviatura para matrix laboratory (laboratorio de matrices). Éste es un programa (y a la vez un lenguaje de programación) especialmente diseñado para la solución numérica de problemas matemáticos como, por ejemplo, sistemas de ecuaciones lineales, sistemas de ecuaciones no lineales, problemas de valores iniciales, etc.
La página web del programa es http://www.mathworks.com/index.html. En ella se encuentran ejemplos de aplicación del programa para la solución de distintos tipos de problemas reales, videos y documentación.
En los dos primeros laboratorios nos familiarizaremos con el uso de este lenguaje.

GENERALIDADES

1. Comenzar MATLAB : escribir `matlab` o `matlab -nodesktop` en un terminal.
2. Salir de MATLAB : `exit` o `quit`.
3. Ayuda en MATLAB : `helpdesk` o `helpwin` o `help <nombre de comando>`.
4. En cualquier momento del trabajo con MATLAB , con el comando `whos` podremos ver las variables en memoria y con el comando `clear all` podremos eliminarlas de memoria.
5. Cuando en MATLAB realizamos una operación y no asignamos el resultado a ninguna variable, la respuesta se asigna a una variable especial llamada `ans`.
6. Si en la ventana de comandos de MATLAB escribimos un comando seguido por `;`, MATLAB no mostrará el resultado de la operación realizada.

En las secciones que continúan veremos ejemplos del trabajo con números reales, vectores y matrices en MATLAB . Los comandos encerrados en recuadros son los que deben ser escritos en la ventana de comandos de MATLAB , teniendo en cuenta que las palabras que siguen a un signo `%` son sólo comentarios para explicar un comando particular y no tienen que ser escritas en la ventana de comandos de MATLAB .

TRABAJO CON NÚMEROS REALES

En MATLAB es posible realizar operaciones aritméticas entre números reales.

```
>> 10.5 + 3.1
```

También podemos, con el comando `=` asignar números reales a variables. En el siguiente recuadro se asigna a la variable `a` el número real $\frac{1}{3}$; a `b`, el valor $\frac{1}{5}$ y a `c`, $\frac{1}{6}$.

```
>> a = 1/3;      % no se muestra resultado de asignación
>> b = 1/5       % se muestra resultado
>> c = 1/6
>> whos          % sólo a, b y c deberían estar en memoria, ¿qué es ans?
>> ans
```

El comando `format` permite cambiar la forma en que el valor de una variable se muestra en pantalla. Tenga siempre presente que con este comando sólo se cambia la forma en que se muestra el valor de la variable, no se cambia el valor de la variable. En MATLAB los números reales se muestran por defecto en formato `short`.

```
>> a              % ver el valor de a con formato por defecto
>> format long    % cambiar formato a long
>> a
>> format rat     % cambiar formato a rat
>> a
>> format short   % cambiar formato a short
>> a
```

En la tabla 1 se encuentran los comandos que se utilizan en MATLAB para comparar dos números reales. El resultado de la comparación será `1` (si la proposición es verdadera) o `0` (si la proposición es falsa). Con los comandos `&&`, `||` y `~` podemos realizar comparaciones múltiples. Ellos representan los operadores lógicos conjunción ($\wedge$), disyunción ($\vee$) y negación ($\sim$) respectivamente.

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

CUADRO 1. Comparaciones entre números reales en MATLAB

`&&`	$\wedge$
`||`	$\vee$
`~`	$\sim$

CUADRO 2. Conectivos lógicos en MATLAB

```
>> b == c              % b igual a c
>> (a > b) && (b > c)  % (a mayor que b) y (b mayor que c)
>> ~(a <= b || a <= c) % negación de:
                       % (a menor o igual que b) o (a menor o igual que c)
```

En el cuadro 3 se muestran los comandos para realizar operaciones aritméticas entre números reales en MATLAB .

`+`	adición
`-`	sustracción
`*`	producto
`/`	división
`^`	potencia

CUADRO 3. Comandos para operaciones aritméticas

Los paréntesis se usan para cambiar el orden en que se realizan las operaciones aritméticas.

```
>> a + b
>> a*b^3          % Primero se calcula potencia cúbica de b
>> (a*b)^3        % Primero se multiplican a y b
```

Muchas de las funciones conocidas ya están incorporadas en MATLAB . Observe que la constante `pi` de MATLAB contiene el valor de π.

```
>> abs(-3/5)      % valor absoluto de un número real

>> format long
>> pi
>> format short

>> cos(pi/2)      % ¿es cero el resultado?
>> sin(pi/4)
```

```
>> d = sqrt(2)    % raíz cuadrada de un número real

>> log2(d)        % logaritmo base 2 de un número real

>> x = 1/6
>> y = exp(x)     % función exponencial
>> format rat
>> log(y)         % logaritmo natural de un número real
>> format short
```

En los computadores personales se utiliza la aritmética de punto flotante para almacenar los números reales. MATLAB representa cada número como *double*, es decir, usando 64 bits consecutivos de memoria. Las constantes `realmax` y `realmin` contienen el mayor y el menor números reales positivos que pueden almacenarse de esta forma. Note que ellos son aproximadamente iguales a 2^{1023} y 2^{-1023} respectivamente.

```
>> realmax
>> 2^1023
>> realmin
>> 2^(-1023)
```

Si el valor absoluto del resultado de una operación aritmética es mayor que `realmax`, ocurrirá *overflow*. Nos daremos cuenta de que esto ha ocurrido porque MATLAB dará `Inf` o `-Inf` como resultados de la operación, en lugar de un número real. El resultado en MATLAB será `Inf` si el resultado real de la operación aritmética es un número positivo mayor que `realmax`, se obtendrá un `-Inf` cuando el resultado real de la operación aritmética sea un número negativo menor que $-$`realmax`.

```
>> x = 2^512
>> x^2           % resultado real es 2^(1024), mayor que realmax
                 % matlab devuelve Inf
>> x = -2^342
>> x^3           % resultado real es -2^(1026), menor que -realmax
                 % matlab devuelve -Inf
```

Si el valor absoluto del resultado de una operación aritmética es menor que `realmin`, ocurre *underflow*. Pero en MATLAB se resuelve este problema de manera desapercibida para el usuario. Note que, por ejemplo, a pesar de que el resultado de la siguiente operación es menor que `realmin`, MATLAB muestra un número real como resultado y no un valor especial como en el caso anterior.

```
>> x = 2^(-512)
>> x^2
```

Si realizamos alguna operación aritmética cuyo resultado no pueda ser determinado, el resultado será `NaN` (*not a number*).

```
>> 1/0            % resultado es Inf
>> 0/0            % resultado es NaN
```

En MATLAB la precisión del computador se denota por `eps`, éste es el menor número real positivo x que satisface que el resultado de la operación $1 + x$ es distinto de 1.

```
>> eps
>> (1+eps)-1      % es distinto de cero
>> (1+eps/2)-1    % es cero
```

Trabajo con vectores y matrices en Matlab

Como su nombre lo indica, Matlab está especialmente diseñado para el trabajo con matrices. Un vector fila x ($x \in M_{1n}(\mathbb{R})$) en Matlab se crea escribiendo cada una de sus componentes, separadas por comas o espacios, entre `[ ]`. Un vector columna x ($x \in M_{n1}(\mathbb{R})$) en Matlab se crea escribiendo sus componentes, separadas por punto y coma, entre `[ ]`.

```
>> x = [1 2 3 4 5]             % vector fila
>> y = [1; 1/2; 1/3; 1/4; 1/5] % vector columna
>> z = 1:5                     % genera un vector igual a x
>> u = ones(5,1)               % crea una matriz de 5 filas y
                               % 1 columna (un vector columna)
                               % cuyas entradas son todas iguales a 1
```

Con los comandos `length` y `size` podemos preguntar la dimensión de un vector. El resultado de `length(x)`, siendo `x` un vector, es el número de componentes de `x`. Con `size(x)` preguntamos el número de filas (primer valor de salida) y de columnas de `x` (segundo valor de salida).

```
>> length(x)    % número de componentes de x
>> size(x)      % número de filas y número de columnas de x
```

Con los mismos comandos que se usan para comparar números reales (ver cuadro 1) podemos comparar dos vectores de igual dimensión, esta comparación se hace componente a componente. El resultado será un vector con la misma dimensión que los vectores que se comparan, sus componentes tomarán los valores `0` o `1`.

```
>> x == z
>> x >= y               % operación inválida
                        % tienen el mismo número de componentes, pero
                        % x es vector fila e y es vector columna.
>> w = x'               % asignar a w la transpuesta de x
                        % w es vector columna
>> w >= y
```

Al igual que con los números reales, también podemos formar proposiciones lógicas compuestas con ayuda de conectivos lógicos. Éstos pueden ser `&`, `|` y `~`, que no son exactamente iguales a los listados en el cuadro 2, y realizan las operaciones de conjunción, disyunción y negación de proposiciones lógicas con vectores componente a componente. Con `all(x)` podremos saber si todas las componentes de un vector son distintas de cero. Con `any(x)` es posible averiguar si al menos una de las componentes de un vector es distinta de cero.

```
>> (w >= y) & (y >= u)       % resultado es un vector
>> all(ans)                  % averiguar si la relación se cumple
                             % para todas las componentes de los vectores
                             % w, y, u.
```

Una vez que hemos creado un vector podemos ver y/o modificar sus componentes con `( )`.

```
>> y(2)                     % elemento en 2da posición de y
>> y([1 3 5])               % componentes 1, 3, 5 de y
>> y(2:4) = 0.5             % modificando componentes 2 a 4 de y
>> y([1 5]) = [-1 1]        % modificando componentes 1 y 5 de y
```

Los comandos `linspace(p,u,N)` y `p:incr:u` nos permiten crear vectores de muchas componentes equidistantes entre sí. Con el primero de ellos se genera un vector fila de `N` componentes equidistantes entre sí, siendo `p` la primera de ellas y `u`, la última. Con el segundo se genera un vector fila cuyo primer elemento es `p`, el último es `u` y la diferencia entre dos elementos consecutivos es `incr`.

```
>> t = 1:.01:3            % el 1er elemento de t es 1,
                          % el último es 3 y
                          % la diferencia entre
                          % 2 elementos consecutivos
                          % de t es 0.01.
>> s = linspace(1,3,201)  % s y t son iguales
```

Se pueden realizar operaciones aritméticas entre vectores (las dimensiones deben ser las correctas para cada operación).

```
>> s1 = x + z          % suma de vectores

>> 2*t                 % multiplicación por escalar

>> x*y                 % producto entre 2 vectores
>> y*x
```

Las operaciones aritméticas también pueden realizarse componente a componente.

```
>> p = u.*y            % producto componente a componente
>> d = u./y            % división componente a componente
>> x.^2                % cada componente al cuadrado
```

Con la función `norm` podemos calcular normas de vectores.

```
>> norm(x)           % norma 2 de x
>> p = 5;
>> norm(x,p)         % norma p de x
>> norm(x,inf)       % norma infinito de x
```

A pesar de que para todo $x \in \mathbb{R}^n$ se cumple $\|x\|_2 = \sqrt{x^t x}$, debemos siempre calcular la norma 2 de un vector usando la función `norm` de MATLAB y no mediante `sqrt(x'*x)`. Si, por ejemplo, $x = 2^{512}\begin{pmatrix}3\\4\end{pmatrix}$, su norma 2 es $5 \cdot 2^{512}$ que es menor que `realmax`. Sin embargo, $x^t x = 9 \cdot 2^{1024} + 16 \cdot 2^{1024}$. Dado que los términos en esta suma son mayores que `realmax`, al calcular `sqrt(x'*x)` ocurrirá *overflow*. En general, en el trabajo con MATLAB , siempre es mejor usar las funciones provistas por el programa que usar expresiones que sean matemáticamente equivalentes.

```
>> x = 2^(512)[3;4]
>> norm(x)
>> sqrt(x'*x)
```

Las funciones sobre números reales disponibles en MATLAB también pueden aplicarse a vectores. Con el comando `plot` podemos graficar estas funciones.
Por ejemplo, para graficar $\sin(x)$ con x entre 0 y 2π,

```
>> x = linspace(0,2*pi,100) % 100 valores entre 0 y 2*pi
>> y = sin(x)               % evaluar la función en x
>> plot(x,y)                % graficar
```

Para crear una matriz escribimos cada una de sus filas separadas por ;. Los elementos dentro de cada fila se separan por comas o espacios y todo se encierra entre `[ ]`.

```
>> A = [1 -1;4 0;1 -1]
>> x = [1 -1];
>> B = [x;4 0;x]         % es igual a A
```

También podemos comparar matrices de igual dimensión

```
>> A == B
```

Para ver y/o modificar las entradas de una matriz usamos `( )`. Con el comando `size` podemos averiguar el número de filas y columnas de una matriz.

```
>> [m,n] = size(A)       % averiguar dimensión de A

>> A(1,2)                % elemento en posición (1,2) de A
>> A(:,2)                % 2da columna de A
>> A(3,:) = [1 0]        % modifica 3ra fila de A

>> A([1 3],2)            % filas 1 y 3, columna 2 de A

>> A = [A [1;1;1]]       % añadiendo columna a A

>> At = A'               % transpuesta de A
```

También pueden realizarse operaciones aritméticas entre matrices, incluyendo operaciones aritméticas componente a componente.

```
>> A + At/2          % primero se multiplica At por 1/2
>> (A+At)/2          % primero se suman las matrices

>> A*At              % producto de matrices
>> A.*At             % producto componente a componente
>> A./At             % división componente a componente
                     % observe la fila 2 de la matriz resultante.
```

Supongamos que $A \in M_n(\mathbb{R})$ es una matriz invertible (su determinante es distinto de cero). Dada una matriz $B \in M_{nm}(\mathbb{R})$, la matriz $X \in M_{nm}(\mathbb{R})$ tal que $AX = B$ es $X = A^{-1}B$. Si $m = 1$, $AX = B$ es un sistema de ecuaciones lineales. La matriz X se encuentra en MATLAB escribiendo `X = A\B`.

```
>> A = [1 -1 1;2 1 2;3 1 1]  % la matriz A es invertible
>> det(A)                    % su determinante no es cero
>> b = [1;1;1]
>> x = A\b                   % es la solución de Ax = b
>> norm(A*x-b)               % comprobando que Ax = b
>> B = [2 1;1 2;1 1]
>> X = A\B                   % es tal que AX = B
>> norm(A*X-B)               % con este comando se puede
                             % calcular la norma 2 de una matriz
>> norm(A*X-B,1)             % y la norma 1
>> norm(A*X-B,inf)           % y la norma infinito
```

Si, en lugar de `A\B`, escribimos `B/A` MATLAB devuelve un mensaje de error pues las matrices A y B de este ejemplo no permiten realizar esta operación. Al escribir `B/A` en MATLAB se pretende calcular BA^{-1}.

Observación:

Recuerden que en la página
http://www.ing-mat.udec.cl/pregrado/asignaturas/521230/documentacion/documentacion.php
pueden encontrar tests, laboratorios y certámenes de años anteriores, así como las láminas del curso.

Índice alfabético