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

Cálculo Numérico (521230/525240)
Laboratorio 1

Introducción a 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://es.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.

Actividad 1: Trabajo con vectores en MATLAB Como su nombre lo indica, MATLAB está especialmente diseñado para el trabajo con matrices.

Un vector fila $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathbb{R}^{1\times n}$) en MATLAB se crea escribiendo cada una de sus componentes, separadas por comas o espacios, entre `[ ]`. Un vector columna $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathbb{R}^{n\times 1}$) 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
>> v= 1./x                      %genera un vector fila resultante de dividir 1
                                %entre cada componente del vector 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
```

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 cuyas componentes son 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 = 0:.01:2           % 0 es 1er elemento de t, 2 es el último
                         % diferencia entre 2 elementos consecutivos
                         % es 0.01.
>> s = linspace(0,2,201)  % 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 = 1;
>> norm(x,p)        % norma p de x
>> norm(x,inf)      % norma infinito de 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
```

Actividad 2: Evaluación de funciones. Escribamos una función en MATLAB que, dado un vector $x \in \mathbb{R}^n,\ n \in \mathbb{N},\ x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$, devuelva el vector $F \in \mathbb{R}^n,\ F = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}$, tal que $F_i = f(x_i),\ i = 1, 2, \ldots, n$ siendo

$$f(x) = \begin{cases} \frac{1-\cos(x)}{x^2}, & x \neq 0, \\ \frac{1}{2}, & x = 0. \end{cases} \tag{1}$$

Antes de escribir esta función necesitamos introducir los ciclos y las condicionales en MATLAB.

Ciclos y condicionales en MATLAB

1. Ciclo `for`: tiene la forma

```
for var = vi:incr:vf
   <instrucciones>
end
```

   Las instrucciones en el cuerpo de este ciclo se ejecutan tantas veces como valores tome la variable `var`, la que se inicializa con `vi` y toma los valores `vi`, `vi+incr`, `vi+incr+incr`, ..., `vf`. Si `incr` es 1, puede escribirse

```
for var = vi:vf
   <instrucciones>
end
```

   Si, queremos, por ejemplo, guardar en un vector `p` los 10 primeros términos de la progresión geométrica (con primer término igual a 1 y razón igual a 3)

$$1, 3, 9, 27, 81, \cdots$$

   podríamos hacerlo con ayuda de un ciclo `for` de la siguiente manera

```
p = zeros(10,1);    % inicializar p
p(1) = 1;           % primer elemento de p es 1
for i = 2:10
   p(i) = 3*p(i-1);
end
```

2. Ciclo `while`: tiene la forma

```
while <condición>
   <instrucciones>
end
```

   Las instrucciones en el cuerpo del ciclo se ejecutan mientras `condición` sea verdadera.
   El vector `p` del ejemplo anterior también puede crearse con un ciclo `while` de la siguiente forma

```
p = zeros(10,1);    % inicializar p
p(1) = 1;           % primer elemento de p es 1
i = 2;
while i <= 10
   p(i) = 3*p(i-1);
   i = i + 1;
end
```

3. Una condicional en MATLAB tiene la forma general

```
if <condición>
   <instrucciones 1>
[elseif
   <instrucciones 2>
else
   <instrucciones 3>]
end
```

donde `condición` es una expresión lógica e `instrucciones` es el conjunto de comandos que se ejecutará si `condición` es verdadera. Los comandos encerrados entre corchetes son opcionales.

Abramos un archivo nuevo en el editor de texto de MATLAB. En el primero escribamos las instrucciones siguientes:

```
function F = mifuncion1(x)
% función para evaluar f(x) = (1-cos(x))/x^2 en cada
% una de las componentes del vector de entrada v

% creando vector de ceros con mismo número de elementos que x
F = zeros(length(x),1);

% ciclo para evaluar f en componentes de x
for i=1:length(x)
   % averiguar si componente i-esima de v es distinta de cero
   if x(i) ~= 0
      F(i) = (1-cos(x(i)))/x(i)^2;
   else
      F(i) = 1/2
   end
end
```

Ésta es la función que permite evaluar $f(x)$. Guardemos este programa con el nombre en `mifuncion1.m`.

Observación: El nombre del archivo donde guardamos la función y el nombre de la función (en 1ra línea del programa) coinciden. Durante todo el semestre mantendremos esta convención.

Regresemos a la ventana de comandos de MATLAB para llamar a `mifuncion1`. Si queremos, por ejemplo, averiguar los valores de f en $0, 0.1, 0.2, 0.3, 0.4, 0.5$, podemos hacerlo mediante

```
>> mifuncion1(0:.1:.5 )   % evaluar f en [0,0.1,0.2,0.3,0.4,0.5]
>> whos                   % ni x ni F (locales a mifuncion1) aparecen
                          % como variables de memoria
```

Evaluemos $f(x)$ en los puntos 0.188×10^{-7}, 0.189×10^{-7}, 0.19×10^{-7} llamando a `mifuncion1`

```
>> mifuncion1([0.188e-7, 0.189e-7, 0.19e-7])
```

Observe que los valores que retorna `mifuncion1.m` son todos mayores que 0.6.

Actividad 3: gráfico de funciones. Escriba un rutero para graficar a la función f en (1) en 100 puntos equidistantes en $[-\pi, \pi]$.

Abramos un archivo nuevo, escribamos:

```
% rutero para graficar f(x) = (1-cos(x))/x^2 entre -pi y pi
x = linspace(-pi,pi);
Fx = mifuncion1(x);
% grafiquemos la función
plot(x, Fx)
```

y guardémoslo en el archivo `plotmifuncion1.m`. Éste es el rutero para graficar $f(x)$. Si escribimos en la ventana de comandos

```
>> help plotmifuncion1
```

MATLAB muestra los primeros comentarios en `plotmifuncion1.m`.

Llamemos al rutero para ver el gráfico de $f(x)$ con $x \in [-\pi, \pi]$

```
>> plotmifuncion1
>> whos              % las variables x y Fx permanecen en memoria
```

Con el gráfico obtenido nos damos cuenta de que la función parece ser siempre menor o igual que $\frac{1}{2}$. Sin embargo, al evaluar f en los puntos 0.188×10^{-7}, 0.189×10^{-7} y 0.19×10^{-7} obtuvimos valores mayores que 0.5. Éste es un ejemplo típico de *cancelación excesiva*, la cual puede ocurrir cuando se restan dos cantidades casi iguales. En la expresión $(1 - \cos(x))/x^2$ si $x \approx 0$, entonces $\cos(x) \approx 1$ y al hacer $1 - \cos(x)$ se estarán restando dos números casi iguales.