UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICA
Miércoles 27, Marzo 2019

Métodos Numéricos 220138

Laboratorio 1

Introducción al Octave – I

Octave o GNU Octave (`https://www.gnu.org/software/octave/`) es un lenguaje de programación y una herramienta de cálculo. La sintaxis de operación en línea de comandos y para la creación de scripts es muy similar a Matlab. Al ser su licencia Licencia pública general de GNU, puede ser compartido y utilizado libremente.

Un comando Octave 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 ";".
```

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

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).
```

Importante: En el ejemplo anterior, al intentar calcular `u*v`, OCTAVE enviará un error:

```
>> u*v
error: operator *: nonconformant arguments (op1 is 1x8, op2 is 1x8)
```

el cual nos indica que la multiplicación carece de sentido dadas las dimensiones de los vectores. Sin embargo, en general OCTAVE (al igual que MATLAB) permite realizar ciertas operaciones con vectores y matrices de con dimensiones distintas (ver *Broadcasting*), por ejemplo

```
>> x = [1 2 3; 4 5 6;  7 8 9];
>> y = [10 20 30];
>> x + y
ans=[11 12 13; 24 25 26; 37 38 39]
```

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, almacenada en la variable IM.
>> TM=M'                     % Transpuesta de la matriz, almacenada en la variable TM.
>> DET=det(M)                % Determinante de la matriz, almacenado en la variable DET.
>> VP=eig(M)                 % Valores propios de la matriz, almacenados en la variable VP.
>> A(2,3)        % Muestra el elemento que esta en la posicion (2,3).
>> A(:,3)        % Muestra la tercera 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.
```

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

Los siguientes comandos permiten construir matrices preestablecidas:

`eye(n)`	Matriz identidad de dimensión `n`×`n`.
`zeros(m,n)`	Matriz de ceros de dimensión `m`×`n`. De forma similar, `zeros(n)` devuelve matriz cuadrada `n`×`n`.
`ones(m,n)`	Matriz de unos de dimensión `m`×`n`. De forma similar, `ones(n)` devuelve matriz cuadrada `n`×`n`.
`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(m,n)`	Matriz de dimensión `m`×`n` generada aleatoriamente con valores en [0, 1). De forma similar, `rand(n)` devuelve r
`hilb(n)`	Matriz de Hilbert de dimensión `n`×`n`.

Podemos combinar los comandos anteriores para ahorrar tiempo (y espacio cuando programemos) 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))
```

Octave permite comparar dos números reales. En la siguiente tabla se encuentran los comandos que Octave utiliza para esta acción. El resultado de la comparación será 1 (si la proposición es verdadera) o 0 (si la proposición es falsa).

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

Los comandos `&&`, `||` y `~=` representan los operadores lógicos conjunción ($\wedge$), disyunción ($\vee$) y negación ($\sim$) respectivamente.

```
>> b == c              % b igual a c
>> (a > b) && (b > c)  % (a mayor que b) y (b mayor que c)
>> ~(a <= b || a <= c) % negacion de:
                       % (a menor o igual que b) o (a menor o igual que c)
>> C=(A ==c)           % Si A es una matriz, entonces C es matriz de valores 0 y 1.
                       %  C(i,j)=1 si A(i,j)=c
```

Importante: Las comparaciones de cadenas de caracteres (*strings*) se realizan con la función `strcmp`, no con los operadores de comparación enumerados anteriormente.

OCTAVE 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') % De esta forma se pueden dibujar dos curvas en un mismo grafico.
```

También se pueden hacer 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')  %
>> subplot(2,2,3) % Estamos usando la tercera ventana.
>> plot(x,y,'*')  %
>> subplot(2,2,4) % Estamos usando la cuarta ventana.
>> plot(x,y,'*y') %
```

MATLAB también permite hacer gráficos de superficies, mediante el comando *surf*. Por ejemplo:

```
>> x= [-8:.1:8];
>> y=x;
>> [xx, yy] = meshgrid (x, y);          % Creamos los puntos (x,y) en los cuales
>>                                      % evaluaremos la funcion.
>> r = sqrt (xx .^ 2 + yy .^ 2) + eps;
>> zz = sin (r) ./ r;                   % funcion z=f(x,y).
>> surf(xx, yy, zz);                    % Graficamos (x,y,f(x,y)).
```

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
```