Facultad de Ciencias Astronómicas y Geofísicas

Universidad Nacional de La Plata

Nociones Básicas de Octave

Autores:
Galván Romina
Villanueva Julieta E.
Soldi Mariangeles
Gende Mauricio

2009

1. ¿Qué es Octave o GNU Octave?

Octave o **GNU Octave** es un programa libre para realizar cálculos numéricos. Como indica su nombre es parte del proyecto GNU. MATLAB es considerado su equivalente comercial. Entre varias características que comparten se puede destacar que ambos ofrecen un intérprete permitiendo ejecutar órdenes en modo interactivo.

El proyecto se creó en el año 1988 con el fin de utilizarlo en un curso de diseño de reactores quimicos. Luego, en el año 1992, se decide extenderlo y comienza su desarrollo a cargo de John W. Eaton. La versión 1.0 apareció el 17 de febrero de 1994.

2. ¿Cómo se utiliza Octave?

Octave se puede utilizar de dos formas:

2.1. Por línea de comando

Una vez que se aseguró que tiene instalado el programa en la computadora que va a trabajar, en una terminal escribe octave e ingresa al programa. De esta forma puede escribir directamente los cálculos que quiere realizar.

Ejemplo:

```
octave:1>1+1
ans =  2
octave:2>
```

2.2. Por archivo .m

Esta forma se utiliza en el caso en que deba elaborar algun programa de cálculo largo, lo que facilitara la comprensión del problema y permite modificar alguno de los parámteros sin tener que escribir por línea de comando de nuevo las variables que dependan del parámetro que modificó.

Para trabajar por archivo usted debe asegurarse de estar parado en el directorio correcto en donde está el archivo y si no lo está, debe darle el camino para llegar al archivo.

Existen dos formas de correr un archivo .m, o escribiendo en una terminal `octave archivo.m`, o ingresando al programa y en la línea de comando escribir `archivo.m`. Tal vez es más útil la segunda opción ya que primero, si es necesario seguir trabajando con las variables del programa, al hacerlo dentro del programa las variables ya estan ingresadas, y segundo, si se trabajan con resultados gráficos y no se tiene cuidado, al correr el programa externamente el gráfico aparecerá y desaparecerá en un instante.

Dentro de Octave se pueden utilizar herramientas del bash, como *ls*, *pwd*, *cd*, etc.

Ejemplo:

Supongamos que queremos correr un archivo llamado *programa1.m* que se encuentre dentro del directorio /home/alumno21/referenciacion/practica2/, entonces para correr el programa podríamos hacerlo de las siguientes formas:

1. Desde afuera

   ```
   alumno21@carina:~$cd ~/referenciacion/practica2/
   alumno21@carina:~/referenciacion/practica2$octave prorama1.m
   ```

2. Desde afuera indicando el camino

   ```
   alumno21@carina:~$octave ~/referenciacion/practica2/programa1.m
   ```

3. Desde adentro

   Primero correr `octave`

   ```
   octave:1>cd ~/referenciacion/practica2/
   octave:2>programa1.m
   ```

3. Operaciones básicas con escalares

Octave opera de manera similar a una calculadora o a la notación que se utiliza al hacer las cuentas en papel. Sólo con una pequeña diferencia en los símbolos y en la sintaxis; ya que primero se escribe la variable que contendrá el resultado y luego la operación.

3.1. Formatos

Para visualizar los escalares en el formato deseado se utiliza la sentencia *format opciones*.

Opciones:

- **short**: Imprime números con 4 dígitos significativos
- **long**: Imprime números con al menos 14 dígitos significativos
- **bank**: Imprime números con un formato fijo de dos lugares luego del punto decimal
- **free** o **none**: Muestra la salida en un formato libre

Ejemplo:

```
octave:1> pi
ans =  3.1416
octave:2> format short
octave:3> pi
ans =  3.1416
octave:4> format long
octave:5> pi
ans =  3.14159265358979
octave:6> format bank
octave:7> pi
ans = 3.14
```

3.2. Operaciones aritméticas

Para empezar hagamos algunas cuentas:

- $1 + 2 * 3$
- $1 + 2/4$
- $2 * 2\hat{}3$
- $2\hat{}(1 + 2)/3$

Se disponen de las siguientes operaciones aritméticas para escalares:

- $x + y$ Operación suma
- $x - y$ Operación resta o sustracción
- $x * y$ Operación multiplicación
- x/y Operación división
- $x\hat{}y$ o $x * *y$ Operación potencia
- $sqrt(x)$ Operación raíz

4. Ingreso de los datos

Los datos se pueden ingresar de dos formas:

- Por medio del teclado

  En modo similar a la notación matemática, Octave asigna valores a las variables a través de las *sentencias declarativas*

  Ejemplo:

```
octave:1> a=235.12
a =  235.12
octave:2> b=14
b =  14
octave:3> a+b
ans =  249.12
```

Las variables se pueden borrar con el comando *clear* nombre.

- Por medio de un archivo

  Ejemplo:

  Supongamos que guardamos en el archivo datos.txt el número 235.12 en la primer línea y el 14 en la segunda. Entonces la forma de ingresar estos datos al programa Octave sería:

```
octave:15> M=load("datos.txt")
M =
   235.120
    14.000
```

  si quisiéramos volver a cargar estos valores dentro de las variables a y b como hicimos en el caso del teclado lo que tendríamos que hacer es:

```
octave:1> a=M(1)
a =  235.12
octave:2> b=M(2)
b =  14
```

5. El operador ";"

En todos los ejemplos vistos anteriormente luego de que ingresamos un número o realizamos una operación aparece abajo el número ingresado o el resultado de la operación. El operador ";"permite evitar que esto suceda. Esto es muy útil en el caso de ingresar tablas gigantes para no llenar la pantalla en la que estamos trabajando. Veamos algunos ejemplos:

```
octave:1> a
a =  235.12
octave:2> a;
octave:3>
octave:3> b
```

```
b =  14
octave:4> b;
octave:5>
octave:5> M
M =
   235.120
    14.000
octave:6>
octave:6> M;
```

6. Operaciones básicas con vectores

Para definir vectores utilizamos los corchetes [ ]. Los elementos de una fila se separan mediante espacios en blanco o comas. Los elementos de una columna se separan por puntos y comas o por nuevas líneas.

Ejemplo:

```
octave:15> A=[ 1 2 3 4 5 6 7 8 9]; % vector fila
octave:16> A=[1,2,3,4,5,6,7,8,9]; % vector fila
octave:17> B=[1;2;3;4]; %vector columna
octave:18> B=[1
> 2
> 3
> 4]; %vector columna
```

6.1. El operador ":"

El operador : se utiliza como una recursión para optimizar la carga de variables en el caso en que se tengan que definir valores con alguna relación consecutiva. La forma de utilizarlo es:

- 1er elemento del vector: incremento: último elemento
- 1er elemento del vector: último elemento (incremento = 1)

Ejemplos:

```
octave:1> 1+[1:5]
ans =
   2   3   4   5   6
octave:2> u = [1:4]
c =
   1   2   3   4
octave:3> v = [5:8]
v =
   5   6   7   8
octave:4> w = u + v
w =
    6    8   10   12
```

IMPORTANTE: Observar que para sumar (+) o restar (-) los vectores deben tener la misma longitud. Una forma de verificar esto antes de realizar el cálculo es con la funcion *length*, que calcula la longitud del vector, o con la función *size*, que entrega como resultado el tamaño del vector (más adelante veremos que se utiliza también para matrices)

6.2. El operador "."

El operador . se aplica delante de los operadores de multiplicación (*), división (/) y potencia (^), en el caso en que se necesiten realizar multiplicaciones o divisiones elemento a elemento entre dos vectores.

Ejemplo:

```
octave:5> q=u.*v
q =
    5   12   21   32

octave:6> r=u./v
r =
   0.20000   0.33333   0.42857   0.50000
```

Operaciones	Operaciones puntuales
+ suma	+ suma
- resta	- resta
* multiplicación	.* multiplicación
/ división	./ división
$\wedge$ potenciación	.$\wedge$ potenciación

Además del producto componente a componente entre dos vectores se pueden realizar otras operaciones. A continuación nombraremos las más importantes.

- **Producto escalar entre dos vectores**: Utiliza la función intrínseca *DOT*, la cual devuelve un número.

  Ejemplo:

```
octave:33> p=dot(u,v)
p =  70
```

- **Producto vectorial entre dos vectores**: Se puede realizar solo en el caso en que los vectores tengan 3 componentes y el mismo devuelve un vector perpendicular a los ingresados, a través de la función intrínseca *CROSS*

  Ejemplo:

```
octave:7> r=[1,0,0];
octave:8> s=[0,1,0];
octave:9> t=cross(r,s)
t =

   0   0   1
```

- **Función transpuesta de un vector**: Para transponer un vector simplemente se debe agregar luego de cerrar el corchete el símbolo '.

  Ejemplo:

```
octave:40> v'
ans =

   5
   6
   7
   8
```

7. Operaciones básicas con matrices

Para ingresar una matriz se escriben los valores de los coeficientes entre corchetes y separados por comas para especificar las columnas en una misma fila, y por punto y coma para ir a la fila siguiente.

Ejemplos:

```
octave:10> a=[1,2;3,4]
a =
   1   2
   3   4

octave:11> b=[6,7;8,9]
b =
    6    7
    8    9
```

Para sumar (+) y restar (-) matrices se trabaja igual que con los escalares, sólo teniendo en cuenta que las dimensiones de las mismas deben coincidir.

Ejemplo:

```
octave:13> c=a+b
c =
    7    9
   11   13
```

En la multiplicación (*) de matrices, hay que tener especial cuidado con las dimensiones para poder operar. Más allá de eso, el procedimiento es análogo, a como se hace por definición.

Ejemplo:

```
octave:14> d=a*b
d =
   22   25
   50   57
```

Caso particular:

```
octave:15> d=a.*b
d =
    6   14
   24   36
```

nos dará como resultado la multiplicación componente a componente de las matrices (que es distinto a como se hace por definición). **Sólo se puede realizar con matrices cuadradas**.

La división, siendo las matrices adecuadas para la operación en cuanto a la dimensión, es de la forma:

```
octave:16> e=a\b
e =
  -4  -5
   5   6
```

que corresponde a calcular:

```
octave:17> e=inv(a)*b
e =
  -4  -5
   5   6
```

Visto desde el álgebra, en este caso lo que se esta haciendo, es resolver un sistema de ecuaciones.

Ejemplo:

Un sistema de 2 ecuaciones con 2 incógnitas; tenemos el problema $\mathbf{AX} = \mathbf{B}$, donde $\mathbf{A}$ es una matriz de 2x2, $\mathbf{X}$ es un vector de 2x1 y $\mathbf{B}$ es otro vector de 2x1. Entonces:

```
A =              X =              B =
   2   3             x                6
   7   5             y               32
```

para hallar las soluciones, es decir, x e y hay dos opciones:

- resolver el sistema

$$\begin{cases} 2x + 3y = 6 \\ 7x + 5y = 32 \end{cases} \tag{1}$$

- lo que es equivalente, calcular **X= inv(A)*B**

  Nota: A eso se llega si multiplicando en ambos lados de la ecuación por la inversa de A:

$$\mathbf{A}.\mathbf{X} = \mathbf{B}$$

$$\mathbf{A^{-1}}.\mathbf{A}.\mathbf{X} = \mathbf{A^{-1}}.\mathbf{B}$$

$$\mathbf{Id}.\mathbf{X} = \mathbf{X} = \mathbf{A^{-1}}.\mathbf{B}$$

El cálculo de la matriz transpuesta se representa de la misma forma que para vectores.

Ejemplo:

```
octave:18> g=a'
g =
   1   3
   2   4
```

7.1. Manipulación de una matriz

7.1.1. Uso del operador ":"

En Octave, se puede tomar a toda una columna o fila de una matriz y redefinirla como un nuevo vector.

```
A =
   2   7
   3   5
octave:19> c1=A(:,1)
c1 =
   2
   3
octave:20> f1=A(1,:)
f1 =
   2   7
```

En el primer caso buscamos todos los elementos de la columna 1, y lo renombramos como el vector c1. En el segundo caso buscamos todos elementos de la fila 1 y lo renombramos como f1.

Este uso del operador es muy útil en el caso de tener que buscar condiciones con respecto a alguna variable específica dentro de una lista de variables, tomadas como una matriz. De esta forma elegimos solo la variable de interés, la renombramos dentro de un vector y luego buscamos sólo dentro de este vector. Veremos algunos ejemplos cuando expliquemos el operador **FIND**.

7.1.2. Ordenar una matriz con el operador SORT

La función intrínseca **SORT** tiene como objetivo ordenar los coeficientes de las columnas de una matriz de manera creciente.

```
octave:21> X=[1,5,12;4,3,13;3,2,14]
X =
    1    5   12
    4    3   13
    3    2   14
octave:22> Y=sort(X)
Y =
    1    2   12
    3    3   13
    4    5   14
```

Dicha función también puede ser usada para producir una matriz que contenga los índices originales (por fila) de la matriz ordenada.

Ejemplo:

```
octave:23> [Y,i]=sort(X)
Y =
    1    2   12
    3    3   13
    4    5   14
i =
   1   3   1
   3   2   2
   2   1   3
```

NO puede ser usada para ordenar una matriz por filas. (Al menos en un solo paso).

Ejemplo: Cómo ordenar una matriz según la segunda columna, pero que a la vez las filas mantengan sus valores?

Para realizar esto deben realizarse varios pasos:

1. ordenar la matriz por la segunda columna

   ```
   octave:9> [Y,i]=sort(X(:,2))
   Y =              i =
      2                 3
      3                 2
      5                 1
   ```

2. volver a escribir la matriz X según los índices

   ```
   octave:10> T=X(i,:)
   T =
       3    2   14
       4    3   13
       1    5   12
   ```

7.1.3. Encontrar condiciones lógicas con la sentencia FIND

La sentencia **find**(*sentencia lógica*) busca cuáles son los índices de un vector que satisfacen dicha sentencia lógica. Para armar la sentencia lógica se utilizan los operadores relacionales. En el siguiente gráfico se muestran alguno de estos operadores:

>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
==	Igual
!=	Distinto

Ejemplo:

Supongamos que tenemos una lista con las coordenadas de un grupo de puntos y queremos analizar que pasa con esos puntos en una zona en particular. Podriamos utilizar la sentencia FIND para filtrar y quedarnos solo con los puntos que caen en la zona de interés.

Supongamos que en la primera columna tenemos la latitud, en la segunda la longitud, y en la tercera velocidad de un auto que pasa por esas coordenadas, y que queremos buscar los puntos que estén entre latmin, latmax, longmin, longmax. Entonces la forma de utilizar la sentencia es:

```
ind=find(M(:,1)>=latmin&M(:,1)<=latmax&M(:,2)>=longmin&M(:,2)<=longmax)
```

Con esta sentencia, hallamos los índices que cumplen tal condición. Ahora podemos utilizar estos índices para evaluar, por ejemplo a que velocidad iba pore las coordenadas que están en la región de interés. La forma de hacerlo es la siguiente

```
vel_int=M(ind,3)
```

8. Control de flujo

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de comandos.

Las condiciones que aparecen en estas estructuras son expresiones que, como todas las de octave, devuelven una matriz.

La condición será cierta si todos los elementos de la matriz resultado son distinto de cero. La condición es falsa si alguno de los elementos son cero.

8.1. IF ELSE

Es una estructura de decisión. Su sintaxis es:

if(condición 1)
lista 1
elseif(condición 2)
lista 2
else
lista 3
endif

Las condiciones no son expresiones ejecutables, sino que tienen el rol de controlar que parte del **if** se debe ejecutar. Es decir, si la primera condición es verdadera entonces se ejecutan las sentencias que haya dentro de la **lista 1**. De lo contrario analiza las condiciones del **elseif**, si es verdadera, ejecutará la **lista 2**; y si también fuera falsa, entonces ejecutará la **lista 3** que corresponde al **else**.

Pueden haber tantos **elseif** como sean necesarios; sin embargo, sólo puede haber un **else**, que debe ir como última opción.

8.2. WHILE

Es una estructura de repetición o bucle. Sintaxis:

while(condición)
cuerpo
endwhile

Este bloque consiste en ejecutar una y otra vez la lista de sentencias que haya en el cuerpo mientras se cumpla la **condición**. Cuando ésta deje de ser válida, la ejecución se detendrá.

8.3. FOR

También consiste en una estructura de repetición. Sintaxis:

for variable=expresión
cuerpo
endfor

A diferencia del **while**, en este bucle se indica el número de iteraciones; por medio de la expresión, la cual puede indicar valor inicial y final (separados por dos puntos), o una expresión propiamente dicha que indique cual es el inicio y el fin. En este último caso las variables utilizadas para la expresión deben estar bien definidas con anterioridad.

Lo que hará el bucle será tomar el primer valor, ejecutar las sentencias presentes en el cuerpo. Salir y preguntarse si el próximo valor que toma la variable pertenece al intervalo que le indica la expresión; si es así lo vuelve a ejecutar y sino saltea el bucle.

Ejemplo:

```
octave:1>x=0;
octave:2>for t=1:5
octave:3>x=x+1
octave:4>endfor
```

9. Guardar los datos por medio de archivos

El comando para guardar las salidas en un archivo es *save*. Si uno obtuvo como resultado un vector o una matriz de nombre `resultado`, entonces la forma de salvarlo es:

```
save -ascii "resultado.txt" resultado
```

10. Funciones

Octave incluye una serie de funciones matemáticas y trigonométricas que nos ayudan a simplificar algunos cálculos. En las siguientes tablas se muestran algunos de ellos

- Funciones trigonométricas

Funciónes	**Descripción**
sin(x) **cos(x)** **tan(x)** **sec(x)** **csc(x)** **ctg(x)**	Funciones trigonométricas ordinarias para x
asin(x) **acos(x)** **atan(x)** **asec(x)** **acsc(x)** **actg(x)** **asec(x)**	Funciones trigonométricas inversas para x
sinh(x) **cosh(x)** **tanh(x)** **sech(x)** **csch(x)** **ctgh(x)**	Funciones trigonométricas hiperbólicas para x
asinh(x) **acosh(x)** **atanh(x)** **asech(x)** **acsch(x)** **actgh(x)**	Funciones trigonométricas hiperbólicas inversas para x

- Otras funciones

Función	Descripción
sqrt(x)	Raíz cuadrada de x
abs(x)	Valor absoluto de x
log(x)	Logaritmo neperiano de x
log2(x)	Logaritmo en base 2 de x
log10(x)	Logaritmo en base 10 de x
exp(x)	e^x
round(x)	Redondeo de x al mas cercano
ceil(x)	Redondeo al entero superior de x
floor(x)	Redondeo al entero inferior de x
fix(x)	Redondeo hacia el entero más cercano a cero
sign(x)	Devuelve 1 para los elementos positivos y 0 para los negativos

- Funciones para construir matrices

Función	Descripción
rand(n,m)	Crea una matriz aleatoria uniformemente distribuida de nxm, si se invoca con un solo escalar crea una matriz aleatoria cuadrada de n elementos
eye(n,m)	Crea una matriz nxm con diagonal 1, si se invoca con un solo escalar crea una matriz identidad cuadrada de n elementos
diag(x)	Si x es un vector, la función crea una matriz cuya diagonal serán los elementos del vector x la cual empezará la k-ésima columna de la matriz creada. Si x es una matriz creará un vector columna con la diagonal que empieza en la k-ésima diag(x,k) columna de la matriz x. Si k es positivo devuelve la super-diagonal, si es negativo devuleve la sub-diagonal. Si se invoca con un solo escalar el valor k por defecto es 0 y devuelve la super-diagonal principal.

zeros(n,m)	Crea una matriz nxm con sus elementos iguales a 0
ones(n,m)	Crea una matriz nxm con sus elementos iguales a 1
linespace(p,q,n)	Crea una matriz de n elementos espaciados uniformemente desde p a q
inv(x)	Calcula la inversa de la matriz cuadrada x
det(x)	Calcula el determinante de la matriz cuadrada x
trace(x)	Suma los elementos de la diagonal principal de la matriz x (traza)
sum(x)	Devuelve la suma de los elementos del vector x o la suma de los elementos de cada columna de la matriz x
prod(x)	Devuelve el producto de los elementos del vector x ó la suma del cuadrado de los elementos de cada columna
max(x)	Devuelve el máximo para cada columna de la matriz el máximo de los elementos del vector x
min(x)	Devuelve el mínimo elemento del vector x o el el mínimo elemento para cada columna de la matriz x
sort(x)	Ordena de menor a mayor los elementos del vector x columna de la matriz x
median(x)	Calcula la mediana del vector x ó la mediana para cada columna de la matriz x
mean(x)	Calcula la media del vector x ó la media para cada columna de la matriz x