Funciones & Estructuras de control

Universidad de Concepción, Chile
Departamento de Geofísica
Programación Científica con Software libre

Primavera, 2011

Contenidos

Utilidades

En la resolución de problemas es de utilidad conocer la mayor cantidad de funciones octave. A continuación muestro algunas.

- **r = mod(x,y)** entrega el resto de la división x/y
- **disp(x)** desplega un array sin imprimir el nombre del array. También desplega texto sin formato. Cadena de caracteres van entrecomilla.

```
disp('pi es igual a') ; disp(exp(1))
```

- **B = reshape(A,m,n)** retorna la matriz $B^{m \times n}$ que contiene los elementos de A agrupados a modo columna

- **path** lista la ruta de todos los directorios de búsqueda en octave
- **addpath('dir')** función que añade directorios al *path*
- **b = input('prompt')** despliega *prompt* esperando entrada del usuario

Octave Scripts

Un script es algo asi como un guión computacional. Si tienes una serie de instrucciones que deseas ejecutar una y otra vez las puedes almacenar en un script. Este es un archivo de texto que contiene instrucciones y es la forma básica de un programa en octave. Cuando corres un script , este tiene el mismo efecto como si tiperas los comandos del archivo de texto línea por línea en la consola de octave. *Octave scripts* son archivos de texto corriente pero con la extensión .m, también llamados *M-files*

Funciones

Mientras los scripts nos permiten escribir programas con instrucciones específicas , las *funciones* definidas por el usuario o **function files** son una herramienta mucho más potente. Estas te permiten definir una tarea específica para luego llamar desde la línea de comando o un script.
Veamos un ejemplo de como se define una función en octave. En este ejemplo, a la función se le pasa un argumento que debe ser un vector y esta devuelve la media de sus componentes.

Ejemplo

```
function y = media(v)
# ayuda :
# la instrucion y = media(v)
# devuelve el promedio
# de las componentes del vector v

y = sum (v) / length (v);
endfunction
```

Funciones

Octave viene con un set de **function files** que se encuentran en : `/usr/share/octave/3.0.5/m`
Es útil manejar esta forma de código, ya que nos permite ahorrar tareas y mejorar la **programación**, es decir, elaborar código en bloques con una funcionalidad específica.

Funciones

La estructura general de una función es :

function [**v1 v2 vN**]=name(**arg1, arg2, argN**)
cuerpo de la función
endfunction
Donde **v1, v2 ...vN** son los valores de retorno, y **arg1, arg2 ...argN** son las entradas de la función.

Estructuras de control

En programación las *estructuras de control* permiten modificar el flujo de ejecución de las instrucciones de un programa.
control de flujo $\Longrightarrow$ procesamiento por lotes

- Un programa puede operar de muchas formas, basado en las codiciones que tu defines
- Una condición puede ser `verdadera` (1) o `falsa` (0)
- Se puede testear una condición usando los operadores : $<, <=, >, >=, ==, \ =$
- Funciones retornan valor númerico como respuesta a un test, por ejemplo `strcmp('compara','string')` retornará 0

Estructuras de control

Condicional IF

Ejecuta un bloque de sentencias si la condición es verdadera.

sintaxis

```
if  condicion 1
   bloque de instrucciones
elseif  condicion 2
   bloque de instrucciones
else
   bloque de instrucciones
end
```

Operadores booleanos
(AND, NOT, OR, XOR)

symbol	meaning	example
==	equal	`if x == y`
~=	not equal	`if x ~= y`
>	greater than	`if x > y`
>=	greater than or equal	`if x >= y`
<	less than	`if x < y`
<=	less than or equal	`if x <= y`
&	AND	`if x == 1 & y > 2`
|	OR	`if x == 1 | y > 2`
~	NOT	`x = ~y`

Ejemplo

```
# se define a, b para luego
# aplicar condicion
a = 0; b = 2;

if a > b
c=3;
else
c=4;
end
```

Estructuras de control
FOR

Repite un bloque de código determinado número de veces dependiendo del rango de valores que tome la variable.

sintaxis

```
for variable = rango
bloque de instrucciones
end
```

Ejemplo1

```
# dominio incial
x  = -20:.1:20;
# generamos funcion seno
y  = sin(x)./x;
# perturbamos la funcion
yp =y + rand(1,length(x));

# suavizamos la funcion a traves de promedios

for i = 2:length(y)-1
y0(i) = 1/3*(y(i-1) + y(i) + y(i+1));
end
```

Estructuras de control
WHILE

Ejecuta bloque de instrucciones mientras se cumpla la condición.

sintaxis

```
while condiciones
bloque de instrucciones
end
```

Ejemplo

```
# dominio incial
x = -20:.1:20;
# generamos funcion seno
y = sin(x)./x;
# perturbamos la funcion
yp =y + rand(1,length(x));

# suavizamos la funcion a traves de promedios

i=2;
while i < length(x)
y0(i) = 1/3*(y(i-1) + y(i) + y(i+1));
i++;
end
```