1. Visualización de datos con Octave

Versión final 1.0-19 agosto 2002

Octave es un poderoso software de cálculo numérico. En este documento explicamos un subconjunto realmente pequeño de sus comandos, centrados en la visualización de datos. Para un estudio más detallado de Octave, recomendamos los apuntes de *"Introducción a los Métodos de la Física Matemática"*, capítulo 4 ("Una breve introducción a Octave/Matlab").

1.1. Comandos básicos

Digamos que tenemos un programa en C++, `interpolacion.cc`, que toma tres puntos (pares de coordenadas (x_i^0, y_i^0), $x_1^0 < x_2^0 < x_3^0$), y construye una curva de interpolación cuadrática en el intervalo $[x_1^0, x_3^0]$ con ellos. La salida del programa es un archivo `curva.dat`, que contiene una lista de pares ordenados:

```
x_1    y_1
x_2    y_2
x_3    y_3
etc.
```

Deseamos ver esta curva de interpolación. Un modo de hacerlo es utilizar Octave. Para ello, primero abrimos Octave, luego cargamos el archivo de datos, y por último graficamos los datos.

Existen tres modos de usar Octave: en modo interactivo o por medio de scripts, y éstos a su vez pueden ser autónomos o no. Cada uno tiene sus propias virtudes y desventajas, y dependerá del usuario cuál es el modo más ventajoso en cada circunstancia particular.

1.1.1. Modo interactivo

Para abrir octave, basta escribir `octave` en la línea de comandos:

```
vmunoz@llacolen:~/cursos/computacion$ octave
GNU Octave, version 2.0.16 (i386-pc-linux-gnu).
Copyright (C) 1996, 1997, 1998, 1999, 2000 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1>
```

Después de la presentación aparece un prompt, y podemos comenzar a introducir comandos uno por uno. Esto es lo que se llama *modo interactivo*. Para ver la curva dada por los datos en `curva.dat`, basta escribir los siguientes comandos uno tras otro en el prompt de Octave:

```
octave:1>load curva.dat;
octave:2>plot(curva(:,1),curva(:,2));
```

La primera línea carga los contenidos de `curva.dat`, leyéndolos como si fueran una matriz llamada `curva`. Así, dada la estructura del archivo de datos, `curva` resulta ser una matriz de $N \times 2$, de modo que las abscisas de la curva están en la primera columna, y las ordenadas en la segunda. El archivo de datos podría llamarse de cualquier manera, y tener cualquier extensión; Octave asignará los datos contenidos en él a una matriz que se llama igual que el archivo, sin extensión. La única restricción es que los datos deben efectivamente estar distribuidos formando una matriz, de N filas por M columnas, con $N, M \geq 1$.

En la segunda línea, el comando `plot` dibuja las ordenadas, contenidas en la segunda columna (`curva(:,2)` significa algo como "tome los elementos de `curva` contenidos en todas las filas, para la columna `2`"), versus las abscisas, contenidas en la primera columna (`curva(:,1)`). En general, si `x`, `y` son vectores, `plot(x,y)` grafica `y` vs. `x`.

Un número arbitrario de líneas de comandos de Octave se pueden colapsar en una sola usando `;` como separador. El ejemplo anterior es equivalente a:

```
octave:1>load curva.dat;plot(curva(:,1),curva(:,2));
```

Para salir de Octave escribimos `quit` en el prompt.

Digamos ahora que queremos, además de la curva, graficar los tres puntos que la generaron. Para ello podemos hacer que nuestro programa `interpolacion.cc` cree un segundo archivo, `puntos.dat`, que tiene los pares ordenados de dichos puntos, en el mismo formato que `curva.dat` (abscisas en la primera columna, ordenadas en la segunda). Entonces damos las siguientes instrucciones en el prompt:

```
octave:1>load curva.dat;
octave:2>plot(curva(:,1),curva(:,2));
octave:3>hold on;
octave:4>load puntos.dat;
octave:5>plot(puntos(:,1),puntos(:,2),'ob');
octave:6>hold off;
```

La instrucción `hold on` permite "congelar" el gráfico actual, permitiendo que todos los gráficos sucesivos se superpongan al primero, en vez de borrarlo que es el comportamiento default. `hold off` "descongela" el gráfico, y el próximo gráfico reemplazará al actual.

Observamos también que el segundo `plot` tiene un argumento adicional. Ésta es una cadena de caracteres que especifica el formato para graficar la familia de puntos especificados. Un carácter indica el tipo de punto con que se graficará: `'o'` para círculos, '.' para puntos, `'x'` para cruces, `'-'` para unir los puntos con líneas. El otro carácter indica el color: `'b'` es azul (*blue*), `'r'` es rojo (*red*), `'k'` es negro (*black*), etc. Ambas especificaciones pueden ir en cualquier orden: `'ob'`, `'bo'`, `'k-'`, etc. Si una o ambas especificaciones no se indican, se adopta el valor default, que es `'-'` para el tipo de línea y `'r'` para el color. Así, `plot(x,y)` dibujará los puntos unidos por una línea roja; `plot(x,y,'b')` hará una línea azul, `plot(x,y,'o')` hará puntos rojos, etc.

Es muy posible que en una misma sesión de Octave queramos cargar dos veces el mismo archivo. Por ejemplo, si corremos el programa en un terminal, y tenemos Octave abierto en

otro para graficar los datos resultantes, podríamos querer correr el programa de nuevo en el primer terminal, y rehacer el gráfico con los datos actualizados. `load curva.dat` dará un error en ese caso, porque la matriz `curva` ya existe y no puede ser reescrita con nuevos datos de un archivo. Para forzar a Octave a reemplazar la matriz con los nuevos datos se debe usar `load -force curva.dat`.

1.1.2. Scripts

Existe otro modo de usar Octave, y es a través de *scripts*, que no son sino archivos con extensión `.m`, en los cuales cada línea contiene una instrucción o instrucciones para Octave, que se ejecutan igual que si fueran dadas en el prompt en modo interactivo.

Así, podemos rehacer el ejemplo de la curva de interpolación creando un archivo de texto, llamado por ejemplo `interpolacion.m` (la extensión `.m` es obligatoria), con los siguientes contenidos:

```
% interpolacion.m
%    Grafica los datos generados por interpolacion.cc

load curva.dat;
plot(curva(:,1),curva(:,2));
hold on;
load puntos.dat;
plot(puntos(:,1),puntos(:,2),'ob');
hold off;
```

Las primeras dos líneas son comentarios. Octave ignora cualquier texto desde un `%` hasta el siguiente cambio de línea. `#` también sirve para introducir comentarios. La tercera línea es una línea en blanco. Podemos poner líneas en blanco a nuestro arbitrio dentro del archivo para hacerlo más claro. Son ignoradas por Octave.

En ocasiones necesitamos escribir una instrucción que es demasiado grande para que quepa en una sola línea. Octave acepta que una misma línea de instrucciones sea separada en dos, siempre que sean conectadas por puntos suspensivos (`...`). Un ejemplo un poco inútil es el siguiente:

```
load ...
curva.dat;
```

y

```
load ...
    curva.dat;
```

son equivalentes a `load curva.dat;`

Supongamos entonces que hemos escrito nuestro archivo `interpolacion.m`, y que está en el directorio `/home/vmunoz/cursos/computacion/`. Invocamos a Octave desde este directorio, y simplemente escribimos en el prompt `interpolacion` (sin extensión). El efecto

de esto es que Octave busca si existe una función interna con ese nombre; como no la encuentra, busca si existe en el directorio actual (o en el *path* predefinido[1]), un archivo `interpolacion.m`, y lo ejecuta.

1.1.3. Scripts autónomos

Una tercera manera de ver nuestros datos es usando scripts autónomos. Para ello, modificamos nuestro script, agregándole al comienzo la línea:

```
#!/usr/bin/octave -q
```

que indica dónde está el ejecutable de Octave (podemos preguntarle al sistema operativo dónde está en la máquina que usamos dando en el prompt del shell la instrucción `which octave`). La opción `-q` es simplemente para evitar que aparezca en pantalla la presentación de Octave (autor, versión, etc.). A continuación damos permiso de ejecución a nuestro archivo:

```
vmunoz@llacolen:~/cursos/computacion$chmod u+x interpolacion.m
```

Ahora el script se puede ejecutar como si fuera un comando más del shell, en la forma

```
vmunoz@llacolen:~/cursos/computacion$./interpolacion.m
```

Más aún, ahora puede ser invocado como un comando del sistema desde el propio programa `interpolacion.cc`, sin necesidad de abrir un terminal aparte para visualizar los datos. Esto se hace con el comando `system`, disponible en `stdlib.h`. Así, si `interpolacion.cc` tiene la forma:

```
#include <fstream.h>
#include <stdlib.h>

int main(){

   ofstream archivo_curva("curva.dat");
   ofstream archivo_puntos("puntos.dat");

// Comandos de calculo de los datos
//   y escritura de archivos de salida
//   curva.dat y puntos.dat

   archivo_curva.close();
   archivo_puntos.close();

   system("./interpolacion.m");
}
```

[1] Ojo: No es la misma variable `PATH` del shell.

será nuestro propio programa el que se encargue de llamar al script y mostrar la visualización de los datos.

El procedimiento anterior tiene la desventaja de que la ventana gráfica se cierra tan pronto como el programa `interpolacion.cc` termina. Por ello, si se llama a un script autónomo, es conveniente poner una pausa para que tengamos tiempo de ver algo. Poniendo al final de `interpolacion.m` el comando `pause(s);` , con `s` un número real o entero, la ejecución de Octave se detendrá durante `s` segundos, y de ese modo alcanzaremos a ver el resultado mientras nuestro programa en C++ se ejecuta.