Programación con Octave, curso de recuperación de la asignatura Documentation

Release 1.0

Santiago Higuera

March 16, 2015

Contents

Contents:

CHAPTER 1

Introducción

1.1 Instalación de Octave 3.8

Durante este curso utilizaremos *Octave* versión 3.8, la última disponible a la fecha de redactar este manual. Es posible que tengas ya instalada otra versión de *Octave* o quizás *Octave-UPM*. Cualquiera de esas aplicaciones te puede servir para seguir el curso, pero te recomendamos la instalación de *Octave* versión 3.8, que trae importantes mejoras respecto de versiones anteriores de Octave.

Los usuarios de *Matlab* que decidan seguir el curso con *Matlab*, no deberían encontrar problemas. Si surgiera algún tipo de incompatibilidad entre los ejercicios del curso y *Matlab* os agradeceremos que nos lo indiquéis, a través del foro, para ir completando una lista de incompatibilidades, que no será grande, y se tratará más bien de pequeños detalles.

En poco tiempo, seguramente antes de acabar este curso, aparecerá la versión 4.0 de Octave, que traerá todavía más mejoras y correcciones de bugs respecto de versiones anteriores. Os tendremos puntuálmente informados a través del foro de novedades de la asignatura.

Para instalar la versión 3.8 de *Octave* hay que seguir las instrucciones que se dan a continuación, dependiendo del sistema operativo que tengáis instalado en vuestro ordenador.

Los que tengan ordenadores **Mac** con sistema operativo **OSX** pueden seguir las instrucciones que se dan en el wiki de Octave [1].

Los usuarios de **Linux** pueden acceder a la página de descagas del portal web de Octave [2] y seguir las instrucciones que se dan allí.

Los usuarios de sistemas operativos **Windows** tienen que utilizar la versión del instalador que se ofrece en la página http://mxeoctave.osuv.de [3] para acceder a todas las características de la versión 3.8. Hay que descargar el instalador y ejecutarlo en nuestro ordenador. También ofrecen una versión portable de *Octave*. Portable se refiere a que no requiere instalación. Se puede copiar en una memoria USB, por ejemplo, y ejecutar en cualquier ordenador con sistema operativo Windows.

[1] http://wiki.octave.org/Octave_for_MacOS_X

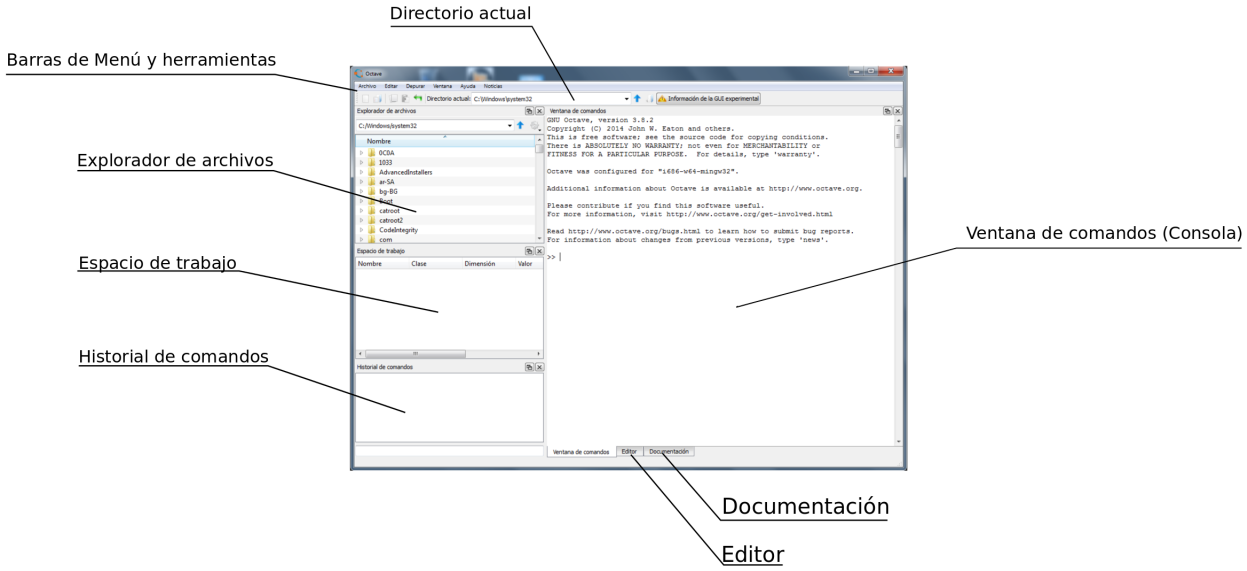
[2] http://www.gnu.org/software/octave/download.html

[3] http://mxeoctave.osuv.de/

1.2 Primeros pasos tras la instalación

Si la instalación ha transcurrido normalmente, al final de la misma deberíamos tener unos accesos directos en el escritorio para poder arrancar *Octave* en modo consola o en modo gráfico. Si arrancamos *Octave* en modo gráfico

accederemos al *Graphic User Interface* de *Octave*, una de las novedades de la versión 3.8 y que se oficializará en la versión 4.0. En la siguiente figura se muestran las diferentes secciones de la ventana gráfica de *Octave*. (Si ha habido fallos en la instalación leed la nota al final del apartado o plantead las dudas en el foro)



En la parte superior tenemos la **barra del menú principal** que permite acceder a distintas opciones del programa y, debajo de ella, la **barra de botones y herramientas**, que permite acceso directo a algunas de estas opciones. Dentro de esta última es importante el desplegable que nos permite conocer o cambiar el **directorio actual de trabajo**.

Debajo de la barra de herramientas aparecen varias ventanas que vamos a describir:

En la parte izquierda tenemos la ventana del **explorador de archivos** que nos ofrece también un desplegable para el **directorio actual** y una ventana que permite acceder al **arbol de directorios** del ordenador.

Debajo del explorador de archivos está la ventana correspondiente al **espacio de trabajo**. En ella irán apareciendo las variables que tengamos en memoria, a medida que vayamos ejecutando comandos en Octave.

Debajo del espacio de trabajo hay una ventana con el **historial de comandos** tecleados desde el inicio de la sesión. Si queremos volver a ejecutar un comando, no tenemos más que hacer doble click con el ratón sobre él y veremos cómo se ejecuta en la ventana de comandos.

La parte derecha de la ventana de *Octave* está dedicada a la **ventana de comandos** o **consola**, que es el lugar destinado a escribir las órdenes que queramos ejecutar, y donde se nos mostrará la salida de resultados. Esta ventana está tabulada por pestañas o solapas(ver parte inferior), que dan acceso a otras dos ventanas: la ventana del **editor** y la ventana de acceso a la **documentación** de *Octave*. El editor es un editor de texto donde podremos crear y modificar los archivos *.m* de nuestros programas Octave. La pestaña de documentación permite acceder a la documentación de *Octave* en modo local (sin necesidad de acceso a internet).

En la *ventana de comandos*, a la que llamaremos también de manera indistinta *consola*, es donde realizaremos la mayor parte del trabajo. Para poder teclear instrucciones hay que hacer que la ventana adquiera el foco de entrada del usuario pinchando sobre ella. A partir de ese momento el cursor parpadeante nos indicará que está esperando que tecleemos alguna instrucción. El funcionamiento es sencillo: se teclea una instrucción más la tecla *Return* , (*Intro*), y *Octave* responde a través de la misma consola con el resultado de intentar realizar el comando tecleado. Si el comando es correcto nos devolverá el resultado, si no, nos devolverá un texto informando del error detectado por el intérprete de lenguaje de *Octave*.

Octave tiene comandos de consola que nos permiten acceder a la información que muestran las ventanas del interface gráfico descritas anteriormente. Es conveniente tener agilidad en la utilización de esos comandos, no limitarse a utilizar las ventanas del interface gráfico.

La ruta del directorio actual de trabajo se puede mostrar en consola con el comando **pwd**. Pruebe a teclear el comando

pwd en la ventana de comandos. Verá que se muestra la ruta del directorio actual y que coincide con la mostrada en las barras desplegables del interface gráfico.

Note: **pwd** es el acrónimo de *print working directory*

Note: **Directorio** es cada una de las divisiones lógicas que se hacen en los sistemas de archivos y que pueden contener otros archivos o directorios. Desde la aparición de los interfaces gráficos para los sistemas operativos se los empezó a denominar también **carpetas** por analogía con las carpetas de los archivos físicos. El icono que se suele utilizar para representar un directorio de un disco es el de una carpeta. En este curso utilizaremos indistintamente las palabras directorio o carpeta para referirnos a ellos. También es habitual la denominación *subdirectorios*, cuando queremos referirnos a los directorios que están contenidos en uno concreto: *'Los subdirectorios del directorio 'programs', que en terminos gráficos serían las carpetas que están contenidas en la carpeta 'programs'*.

El contenido del directorio actual lo podemos mostrar en la consola tecleando el comando **dir** o el comando **ls**, según prefiramos el estilo *windows* o el estilo *linux*. Pruebe una vez más a teclear el comando en la ventana de comandos y a comprobar que el contenido de carpetas y archivos que muestra corresponde al contenido del directorio actual mostrado por la ventana **Explorador de archivos**.

Podemos cambiar el **directorio de trabajo** desde la consola tecleando el comando **cd**, (*change directory*), que admite varias formas de uso:

- **cd nombre_directorio**: Cambia el directorio actual al directorio de nombre *nombre_directorio*, que debe ser uno de los subdirectorios del directorio actual.
- **cd ruta_completa**: Cambia el directorio de trabajo actual a uno especificado por su ruta completa.
- **cd ..**: (*cd punto punto*)Cambia al directorio *padre* del directorio actual. El directorio *padre* es el directorio que contiene al directorio actual.

Tip: Pruebe a teclear varios cambios de directorios a través de la consola. Conviene ejercitarse en el manejo de la consola para moverse por los directorios del ordenador. No solo es eficiente, sino que además es imprescindible en determinados entornos de trabajo en los que no es posible disponer de interface gráfico para Octave. También es importante saber manejar nombres de archivos y sus rutas de acceso a la hora de trabajar en *lectura-escritura* de ficheros. A medida que teclea los comandos de cambio de directorio, observe como cambia el contenido de los desplegables del directorio actual y de la ventana del Explorador de archivos.

Cada vez que teclea un comando en la consola de *Octave* se puede observar como se actualiza el contenido de la ventana **historial de comandos**. En esta ventana se muestran de manera ordenada los distintos comandos tecleados con anterioridad. Se puede volver a ejecutar un comando haciendo doble click sobre él. Para acceder al historial de comandos desde la consola tenemos que teclear el comando **history**, que mostrará un listado de los comandos tecleados con anterioridad, con un número de comando asignado a cada uno de ellos. Podemos ejecutar un comando de la lista tecleando **run_history number**, donde *number* es el número de comando en el listado *history*. Podemos ejecutar los comandos comprendidos entre dos números haciendo **run_history num1 num2**, que ejecutará todos los comandos entre el comando *num1* y el comando *num2*.

Para comprobar el funcionamiento de la ventana del **espacio de trabajo** hay que definir alguna variable. Pruebe a teclear alguna sentencia que asigne valor a alguna variable, por ejemplo, **a=3.0** o **b=-1**. Verá como tras teclear cada una de las asignaciones anteriores aparecen lineas de información en la ventana **espacio de trabajo**. Eso es lo que denominamos *espacio de trabajo*, el conjunto de variables existentes y guardadas en memoria a cuyos valores podemos acceder en cada momento. Para acceder al espacio de trabajo desde la consola hay que teclear el comando **whos**.

Tip: El comando *whos* nos muestra un listado por consola con las distintas variables existentes y sus tipos de datos. Compruebe, al teclear el comando, que la información mostrada por el comando *whos* se corresponde con la información mostrada en la ventana *espacio de trabajo*.

Podemos borrar una variable del *espacio de trabajo*, o lo que es lo mismo borrarla de memoria, mediante el uso del comando **clear nombre_variable**. Pruebe el comando *clear* borrando alguna de las variables que haya definido anteriormente. Compruebe mediante el comando *whos* y mediante la inspección visual de la ventana del *espacio de trabajo* que la variable borrada ha dejado de estar en memoria. El comando **clear** sin argumentos borrará de la memoria todas las variables que tuviéramos definidas hasta el momento.

Tip: El comando **clc**, (*clear console*), permite borrar la pantalla de la consola y devolver el cursor a la parte superior izquierda. Este comando no afecta la contenido de las variables, y es un comando que también utilizaremos a menudo.

1.3 Obtención de ayuda

La pestaña de documentación situada en la parte inferior de la ventana de trabajo nos da acceso a una documentación *off-line*, (off-line=sin necesidad de conexión a internet). Podemos navegar por las distintas secciones y comandos lo que nos permitirá aprender muchas de las características de *Octave* y del lenguaje *m*.

Para acceder a la ayuda desde la consola disponemos de dos comandos muy útiles: el comando **help** y el comando **lookfor**. El comando *help* es de utilidad cuando conocemos el nombre exacto de la función o comando que queremos consultar. Por ejemplo, si tecleamos **help run_history** podremos ver las distintas opciones que ofrece el comando *run_history*. El comando **lookfor**, en cambio, nos devuelve una lista de funciones y comandos que contengan en su documentación la palabra buscada. Podemos hacer una prueba tecleando el comando **lookfor history** y viendo el listado de funciones y comandos que tienen relación con la palabra *history*.

Tip: El libro '*Matlab y matemática computacional*' de Sagrario Lantarón Sánchez y Bernardo Llanas Juárez sirve de soporte de la asignatura y contiene toda la materia *preguntable* en examen, siendo por ello el primer material de referencia a consultar.

Hay mucha documentación disponible en la red acerca de la utilización de *Octave* y *Matlab*. En general, la mayoría de funciones y comandos funcionan igual en *Octave* o en *Matlab*, por lo que se puede utilizar indistintamente la documentación de un programa u otro.

La mejor documentación de *Octave* es la que hay disponible en la web de Octave y que se actualiza con frecuencia [1]. La mejor documentación de *Matlab* es la de la web de Matlab y conviene tenerla siempre en cuenta a la hora de aprender el uso de funciones y comandos [2].

A lo largo del curso iremos indicando algunos portales y documentos que contienen también información interesante para profundizar en el conocimiento de Octave y Matlab.

-[1] http://www.gnu.org/software/octave/doc/interpreter/

-[2] http://es.mathworks.com/help/matlab/

1.4 Problemas de instalación

Note: Si tras la instalación no se tienen accesos directos en el escritorio para arrancar *Octave* se puede intentar lo siguiente:

- **Localizar el directorio bin de la instalación:** Lo primero que tenemos que hacer es localizar el directorio donde ha quedado instalado el programa. Durante la instalación se nos ofrece instalar en algo parecido a **C:\Octave\Octave-3.8.1**. En general será algo parecido a *C:\Programs\Octave* o *C:\Octave* o también *C:\Archivos de Programa\Octave*. En cualquier caso, dentro del directorio de la instalación debemos localizar el directorio **bin**, que es donde se encuentran los programas ejecutables de *Octave*.

- **Añadir la ruta del directorio bin a la variable path del sistema:** La variable *path* del sistema guarda una lista de directorios separados por *';'*. Estos directorios es donde el sistema operativo busca cada vez que tiene que ejecutar un programa. En el siguiente enlace: Cómo cambiar el path de Windows explican como hacer el cambio. El cambio que hay que hacer es añadir al final de la cadena *path* existente un *';'* y la ruta del directorio *bin* que será algo parecido a *C:\Octave\Octave-3.8\bin*. (ver Nota 1)
- **Poner en el escritorio accesos directos a *Octave*:** Tenemos que prestar atención a los archivos **octave.exe** y **octave-gui.exe**. Son los programas que arrancan *Octave* en modo consola y en modo gráfico, respectivamente. (*GUI es el acrónimo de Graphic User Interface*). Podemos crear unos accesos directos a ambos programas y tenerlos disponibles en el escritorio, de forma que podamos acceder rapidamente a *Octave*.

CHAPTER 2

Semana 1

2.1 Teoría de programación: Cuestión de arquitectura

2.1.1 Los lenguajes de programación y sus tipos

Un lenguaje de programación es un lenguaje formal creado para transmitir instrucciones a un ordenador. Hay numerosos lenguajes de programación de ordenadores. Los lenguajes proporcionan conceptos que permiten razonar en términos abstractos sobre los problemas a resolver. Hay diferentes formas de abordar el problema de transmitir instrucciones a un ordenador. A lo largo de la historia de la programación han ido surgiendo diferentes modelos de programación, plasmados en lenguajes de programación concretos y que han dado lugar a una división de los lenguajes de programación según el modelo que utilizan. A dicho modelo de programación se le suele denominar el *paradigma* de programación utilizado por el lenguaje.

En los **lenguajes declarativos** no se le dice al ordenador cómo resolver el problema, lo que se hace es describir el problema, siendo una instancia posterior quién utilizará esa información para calcular la solución. Un ejemplo de este tipo de lenguajes son los lenguajes utilizados para describir documentos, por ejemplo. Es el caso del *HTML* y del *XML*. Se describe el documento y el ordenador utilizará algún tipo de interprete del lenguaje para crear el documento en cuestión. El ejemplo que se muestra a continuación es el de un documento muy simple creado en lenguaje *HTML*:

```
<html>
    <head>
        <title>Ejemplo</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <h1>Esto es una cabecera</h1>
        <h3>Esto es un título menor</h3>
        <p>Esto es un párrafo de texto</p>
    </body>
</html>
```

Mostramos a continuación cómo se interpreta ese código al utilizar el navegador *Chrome*. Las páginas WEB que acostumbramos a utilizar están escritas en lenguaje *HTML*. Los navegadores que utilizamos son intérpretes del lenguaje

HTML.

Estas líneas que estás leyendo, están escritas en el lenguaje Sphinx, que es un lenguaje declarativo pensado para escribir este tipo de documentos. El interprete del lenguaje lo convierte posteriormente en código *HTML*, que probablemente es la versión que estás leyendo, aunque podría ser también que estuvieras leyendo la versión *pdf* generada por el interprete que hay en el portal donde está colgada esta documentación, y que permite transcribir los documentos a formato *pdf*, que es a su vez otro lenguaje declarativo para describir documentos.

Sin duda, el rey de los lenguajes declarativos para definir documentos es **LaTeX**. Se trata de un lenguaje destinado a escribir documentos, fundamentálmente científicos, que proporciona un tratamiento tipógrafico de alta calidad y un editor de fórmulas incomparable. *LaTeX* proporciona, además, herramientas que permiten gestionar de manera muy eficiente la estructuración del documento y la gestión de referencias bibliográficas, índices y tablas. El concepto de trabajo al escribir documentos con *LaTeX* es diferente al de los procesadores de texto habituales, permitiendo centrarse en el contenido del documento y no en su formato o presentación.

Existen otros lenguajes declarativos que no están destinados a describir documentos o juegos de datos, sino a obtener la solución de problemas concretos de computación. Merece la pena destacar aquí el *SQL*, lenguaje comúnmente utilizado en el acceso a las bases de datos. Otro tipo particular de lenguajes declarativos son los lenguajes de programación *funcional*, como *Lisp*, *Haskell* o *Scala*.

Por otra parte están los **lenguajes imperativos**, que, en lugar de limitarse a describir lo que quieren obtener como hacen los lenguajes declarativos, detallan las instrucciones que tiene que realizar el computador en cada momento para llegar a la solución del problema a resolver.

Dentro de los lenguajes del tipo *imperativo*, un grupo importante lo constituyen los lenguajes basados en la **programación estructurada**. La *programación estructurada* es un paradigma de programación que divide los programas en rutinas y subrutinas, y que utiliza solamente tres estructuras: secuencia, selección (if y switch) e iteración (bucles for

y while). Dentro de la programación estructurada es donde se encuadra el lenguaje *m* utilizado por **Octave** y **Matlab**.

Otro grupo muy importante de lenguajes de programación imperativa, (otro paradigma de programación), lo constituyen los lenguajes de **programación orientada a objetos**. En estos lenguajes se definen los denominados *objetos* que tienen propiedades y métodos mediante los cuales pueden ser manipulados. Dentro de la programación orientada a objetos destacan el lenguaje *C++* (se lée *C plus plus*) y el lenguaje *Java*.

Existen también lenguajes que pueden ser utilizados bajo distintos paradigmas de programación, lenguajes *multiparadigma*. En general, casi todos los lenguajes permiten formas de utilización mixtas entre unos paradigmas y otros, si bien suelen estar más pensados para un paradigma de programación concreto. Dentro de los lenguajes *multiparadigma* vamos a destacar aquí el lenguaje *python*, que permite abordar los programas desde un punto de vista de programación estruturada, como programación orientada a objetos o como programación funcional.

2.1.2 Arquitectura de un programa

Los programas suelen estar divididos en diferentes unidades lógicas que interactuan y dependen unas de otras. A cada una de estas unidades lógicas en que subdividimos un programa se le suele llamar **módulo**. Según el tamaño y complejidad del programa, los módulos lógicos se podrán corresponder con un fichero concreto, o con un directorio completo repleto de ficheros, o en el extremo contrario, los módulos podrían ser cada una de las rutinas existentes en el único fichero de cierto programa, o cada una de las secciones diferenciadas de una función.

La estructura de módulos que componen un programa, junto con las relaciones y dependencias que se establecen entre ellos es lo que se denomina la *arquitectura del programa*. Veamos un ejemplo, pongamos que tenemos un programa que calcula el momento flector en una viga. Supongamos que nuestro programa utiliza unas rutinas, desarrolladas anteriormente, que permiten calcular el momento flector cuando la viga es biapoyada y otro programa que calcula el momento flector cuando la viga es empotrada. Esto hace que nuestro programa *dependa* del *módulo de cálculo de la viga biapoyada* y del *módulo de cálculo de la viga empotrada*. A su vez nuestro programa podría constar de varios módulos: un módulo principal que proporciona el arranque y la estructura general del programa, otro módulo encargado de acceder y utilizar el módulo de la viga apoyada, otro módulo que permite acceder al módulo de la viga empotrada, un módulo que permite imprimir los resultados y un módulo encargado de hacer un gráfico con el resultado. Si el programa lo estuvieramos desarrollando en Octave, cada uno de estos módulos se podría corresponder con una función desarrollada en un fichero *.m*.

2.1.3 Aquitectura en programación estructurada

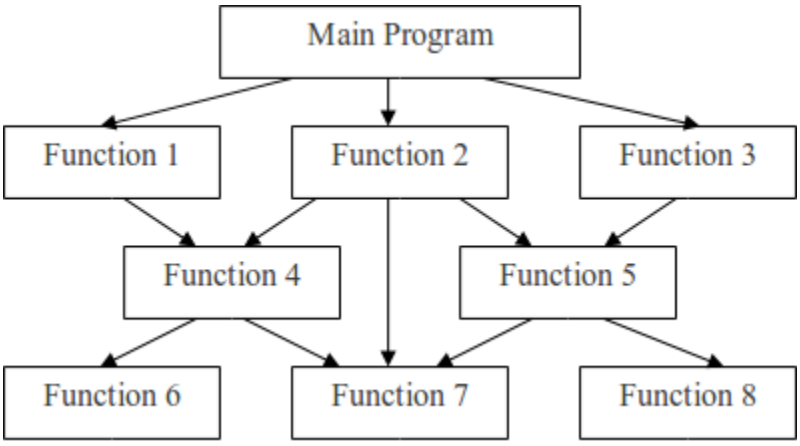
En la programación estructurada los programas están construidos a base de subrutinas. El equivalente en *Octave* de las subrutinas son las funciones.

La arquitectura más simple sería la de un programa que constara de una única subrutina. En *Octave* se trataría de un programa compuesto por una única función y un único fichero *.m*. Es habitual que los programas consten de varias funciones repartidas en uno o más ficheros y que dependen unas de otras. La dependencia hay que entenderla en el sentido de que entre las instrucciones utilizadas por algunas de estas funciones están llamadas a las otras funciones.

Vamos a tratar de explicar la dependencia entre funciones con un ejemplo. Supongamos que desarrollamos una función que permite calcular la solución de un sistema de ecuaciones y que dicha función utiliza, entre sus instrucciones, la llamada a otra función que calcula la inversa de una matriz. Nuestra función utiliza, y por tanto depende de, la función que calcula la inversa de una matriz. Si la función *inversa()* cambia o falla, nuestra función se verá afectada. De hecho, las funciones predefinidas que utilizamos en los programas de *Octave* son funciones desarrolladas por otros y cuyos ficheros *.m* están guardados en algún lugar de la instalación de *Octave*. Nuestras programas *dependen* de ellas, en tanto las utilicen.

Hay algunos principios generales que deben cumplir las dependencias entre módulos en la programación estructurada. Por ejemplo, no conviene establecer relaciones cíclicas entre módulos. No debe suceder que el módulo *A* dependa del módulo *B*, que a su vez depende del módulo *A*. Puede darse una relación cíclica con más intervinientes: El módulo *A* depende de *B*, quien a su vez depende del módulo *C* que depende del módulo *A*. Estas relaciones cíclicas pueden

dar lugar a errores en el desarrollo de los programas y, en cualquier caso, llevan a arquitecturas difíciles de expandir o mantener.



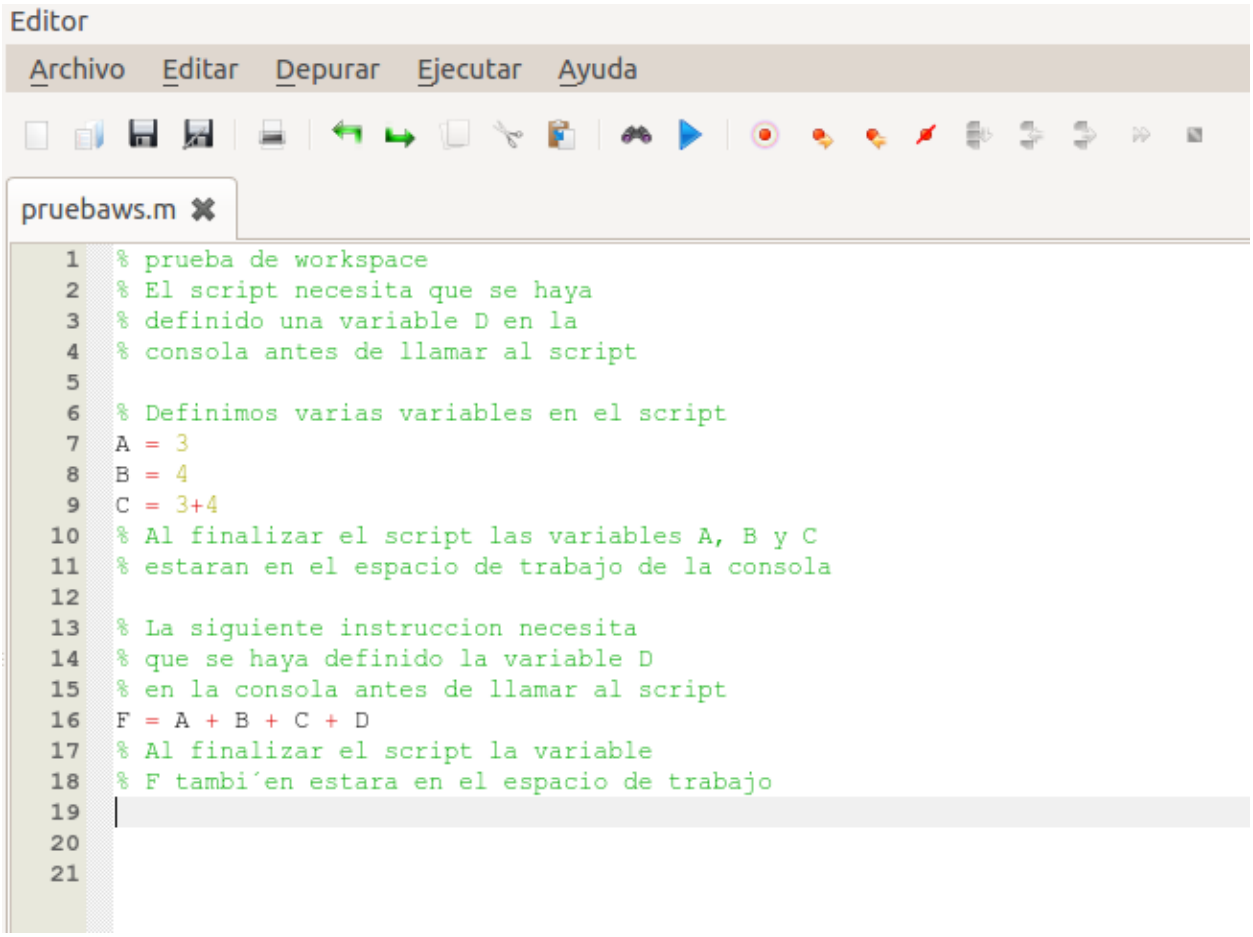
2.1.4 Scripts y Funciones en Octave

Durante el curso aprendemos a utilizar tres maneras de interactuar con Octave: la consola, los *scripts* y las funciones.

La manera más inmediata de operar Octave es a través de la **consola**. El funcionamiento es del tipo *petición-respuesta* habitual en los entornos de consola. El usuario teclea una instrucción válida de Octave en la consola, pulsa la tecla *return* y Octave muestra el resultado de la ejecución de la instrucción en la consola, o un mensaje de error si es que se ha producido alguno. Esto permite utilizar Octave como una calculadora científica avanzada y es uno de los puntos fuertes de Octave de cara a su utilización por ingenieros y técnicos en general.

Otra forma de operar Octave es a través de los denominados **scripts**. La palabra *script* se traduce al castellano por *guión*, aunque en el ámbito de la programación es habitual utilizar la palabra original en inglés. Se trata de una serie de instrucciones dadas al ordenador y que queremos ejecutar una tras otra. Lo que se hace es crear un fichero *.m* y escribir las instrucciones que queremos que ejecute Octave. Para ejecutar un *script* desde la consola de Octave, tendremos que situarnos en el directorio donde está guardado el fichero *.m* del script, y teclear el nombre del fichero (sin la extensión *.m*). El resultado será que Octave irá ejecutando, una a una, las instrucciones contenidas en el fichero de script.

Ejecutar un script es equivalente a ir ejecutando en la consola, una a una, las instrucciones que aparecen en el fichero *.m* del script. Esto implica que el *espacio de trabajo* del script, (las variables activas en memoria en cada momento), es el mismo que el de la consola. Si definimos una variable en el script, cuando termine la ejecución del mismo la variable estará en el espacio de trabajo. Análogamente, si tenemos una variable definida en el espacio de trabajo de la consola, podremos acceder a ella desde dentro del script.



Note: Tip: La barra de herramientas de la ventana del editor de Octave tiene un botón para ejecutar el programa activo. Si pulsamos y pasamos a la ventana de la consola de comandos veremos que lo que hace es ejecutar el programa por consola. Los resultados que se obtienen o los mensajes de error que pudieran aparecer se mostrarán por la ventana de la consola. El comando de ejecución pasará a formar parte del historial de comandos y podremos acceder a él a través del comando *history* o de la tecla de cursor hacia arriba.

Note: Tip: Cuando tecleamos un programa en la ventana del editor de Octave y pulsamos ejecutar, si no estamos en el directorio del fichero *.m*, nos aparecerá una ventana ofreciéndonos cambiar el directorio de trabajo al directorio donde hayamos guardado el fichero *.m*.

La funciones, las **function**, son una forma más sofisticada de programar Octave. Se trata también de ficheros *.m* que contienen instrucciones de Octave, pero en este caso permiten programarse para recibir una serie de parámetros y devolver resultados. En el caso de las funciones, el espacio de trabajo de la función (sus variables activas en memoria en cada momento), no es el mismo que el espacio de trabajo de la consola. Si definimos una variable dentro de la función, cuando salgamos de la función la variable deja de existir. De la misma manera, si tenemos variables definidas en memoria en la consola desde la que llamamos a la función, una vez dentro de la función no tendremos acceso a dichas variables.

El **nombre del fichero** es otro detalle importante a la hora de escribir los ficheros *.m* que guardan los scripts o las funciones. En el caso de los scripts no requieren nada especial, hay que utilizar para el script un nombre de fichero válido acabado en *.m*. En el caso de las funciones, en cambio, el nombre de la primera función que haya en el fichero tiene que coincidir con el nombre del fichero, sin la extensión *.m*.

El formato de los ficheros también es un poco diferente. En el caso de los scripts no hay ningún requerimiento especial,

simplemente iremos escribiendo en el fichero *.m* instrucciones válidas de Octave en líneas sucesivas, en el orden que queramos que se ejecuten. En las funciones es obligatorio que la primera instrucción del fichero diferente de un comentario, sea la denominada *signatura* de la función.

La **signatura** de las funciones tiene que ser la primera instrucción, diferente de un comentario, en el fichero *.m* de la función. Consiste en la palabra **function**, seguida de la lista de valores devueltos entre corchetes, el signo igual, el nombre de la función y, entre paréntesis, la lista de parametros solicitados por la función y separados por comas. Es más difícil describirlo que escribirlo:

```
function [resultado1, resultado2] = nombreDeFuncion(parametro1, parametro2, parametro3)
```

Para utilizar la función, para llamarla, se utiliza la misma sintaxis que la de la *signatura* pero sin la palabra *function* al principio. Veamos un ejemplo de utilización de una función. La función *size()* predefinida de Octave, recibe como parámetro una matriz o vector y devuelve el número de filas y columnas que tiene. La signatura de la función es :

```
[filas, columnas] = size(M)
```

Para utilizar la función *size()* desde la consola, primero definimos una matriz y luego llamamos a la función, pasándole dicha matriz como argumento:

```
octave:7> A = [0,1,2,3; 4,5,6,7; 8,9,10,11]
A =

    0    1    2    3
    4    5    6    7
    8    9   10   11

octave:8> [m,n] = size(A)
m =  3
n =  4
octave:9>
```

Hay que observar un par de detalles en el código anterior:

- El nombre de la variable que le pasamos a la función no tiene por qué coincidir con el nombre asignado a dicha variable en la signatura. Vemos que el parámetro que hemos pasado a la función es una variable llamada *A*, no *M* como pone en la signatura de la función. Eso da igual, la función espera recibir una variable del tipo matriz, da igual cómo se llame.
- El nombre de las variables en las que recibimos los valores devueltos por la función no tiene por qué coincidir con el nombre de la signatura. Nosotros hemos dicho a la función que nos devuelva los valores en unas variables que hemos llamado *m* y *n*, no *filas* y *columnas* como pone en la signatura de la función. Da igual, la función devuelve dos valores y nos los devolverá en las dos variables que pongamos entre corchetes, no importa el nombre que tengan.

Los parámetros que pasamos a las funciones no tienen por qué ser variables, también pueden pasarse directamente los valores numéricos de los parámetros. También podemos pasar una operación que dé lugar al resultado deseado. Veamos un ejemplo utilizando la función predefinida *sind()* que recibe un ángulo en grados y devuelve el valor del

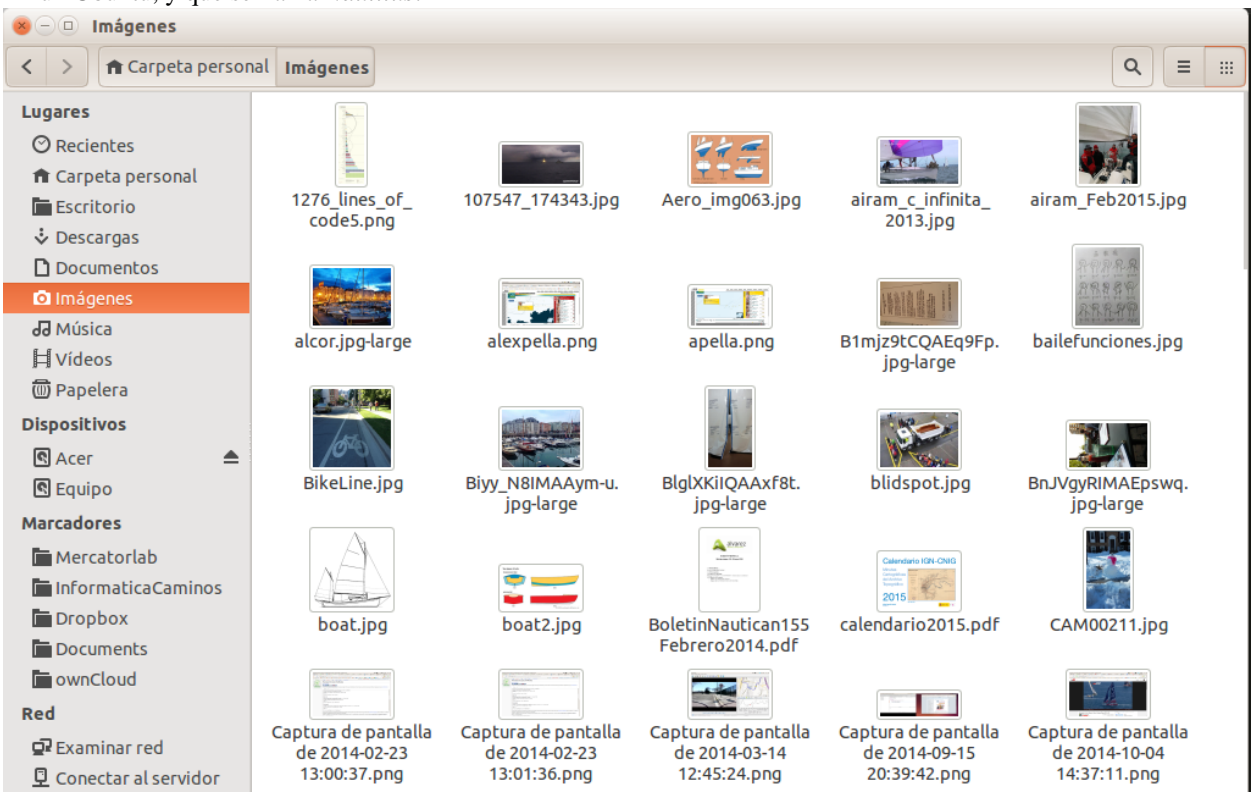
seno de dicho ángulo (*sind* es *sin degrees*, por contraposición a la función *sin()* que recibe el parámetro del ángulo en radianes). En el ejemplo se utiliza también la función *sin()* pasándole una operación basada en la constante predefinida *pi* de Octave.

```
octave:5>
octave:5> a = sind(45)
a =  0.70711
octave:6>
octave:6> b = sin(pi/4)
b =  0.70711
octave:7>
```

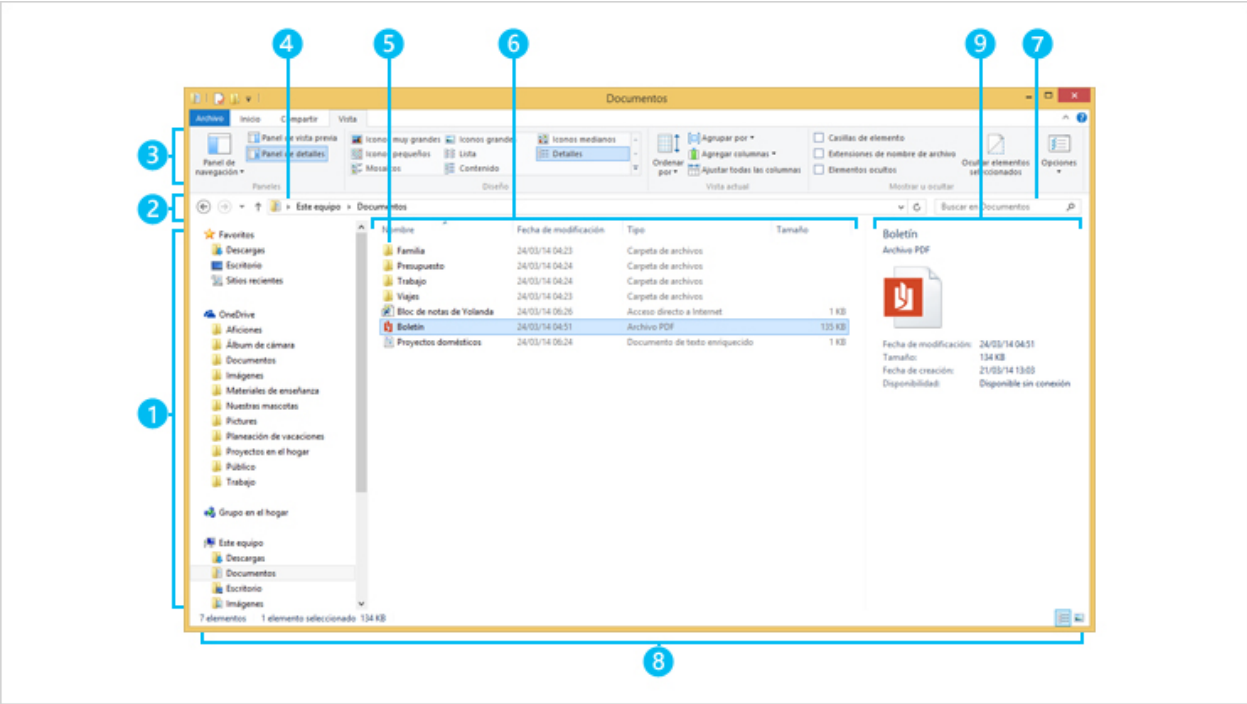
2.2 Tips del sistema operativo 1: el explorador y los nombres de los archivos

2.2.1 El explorador de archivos

La labor de programar y la utilización del ordenador en general, salvo el caso del usuario ocasional, requieren de un manejo intensivo del sistema de archivos. Hay que saber cómo crear, modificar, mover o eliminar archivos y carpetas. Conviene conocer las carpetas que el sistema operativo destina a determinados fines concretos y en qué lugares conviene o no conviene guardar o modificar las cosas. Para todas estas tareas es habitual utilizar el explorador de archivos. En la siguiente imagen mostramos el explorador de archivos que viene por defecto en las distribucioness Linux Ubuntu, y que se llama *Nautilus*:



En esta otra imagen se muestra el explorador de archivos que viene con las últimas versiones de Windows:



Para saber más acerca de cómo manejar el explorador de archivos de la versión 8 de Microsoft Windows puedes consultar el siguiente enlace:

http://windows.microsoft.com/es-es/windows-8/files-folders-windows-explorer

Existen muchas alternativas al explorador que viene por defecto en Windows. En el siguiente artículo tienes 20 alternativas al explorador de archivos:

http://www.emezeta.com/articulos/alternativas-al-explorador-de-windows

Los usuarios de sistemas Linux también disponen de numerosas alternativas. En el siguiente artículo se muestran 14 de ellas:

http://www.muylinux.com/2009/01/07/14-exploradores-de-ficheros-para-linux

Otra alternativa interesante es acceder al sistema de archivos a través de la consola, lo que trataremos en otro momento.

2.2.2 Nombres de archivos y extensiones de nombre

En el sistema operativo Windows la extensión del nombre de archivo, los tres caracteres finales tras el punto, determina el tipo de archivo y lo que se puede hacer con él. Hay archivos **ejecutables** que son los que al abrirlos arrancan un programa. Estos archivos tienen la extensión **.exe**. Los archivos ejecutables suelen tener una serie de archivos asociados que contienen las librerías de procedimientos que utilizan, lo que en otro lugar hemos llamado sus *dependencias*. Estas librerías de procedimientos suelen llevar la extensión **.dll**, (Dynamic Link Library, Librería de Enlace Dinámico). Otros archivos tienen otras extensiones. Los archivos de scripts y funciones de Octave tienen la extensión **.m**.

En general, el sistema operativo Windows asocia una determinada extensión de archivo con los programas que utilizan ese tipo de archivos. También asocia a cada extensión un icono, que es el que sale en el explorador. Estas asociaciones las podemos configurar o modificar. Para saber más acerca de los nombres de archivos y de las extensiones de nombre puedes consultar la documentación de Microsoft en el siguiente enlace:

http://windows.microsoft.com/es-es/windows/file-names-extensions-faq#1TC=windows-7

Note: El explorador de Windows tiene activada por defecto una opción que oculta la extensión de los archivos que llama *conocidos* al mostrarlos en el explorador. Es conveniente desactivar esa opción, y que podamos ver las extensiones de los nombres de archivo listadas. En el siguiente enlace puedes consultar cómo modificar el comportamiento por defecto del explorador de archivos de Windows:

http://windows.microsoft.com/es-es/windows/show-hide-file-name-extensions#show-hide-file-name-extensions=windows-7

2.3 Tips Octave 1: el historial, atajos de teclado y más

2.3.1 Acceder a los comandos del historial con el cursor

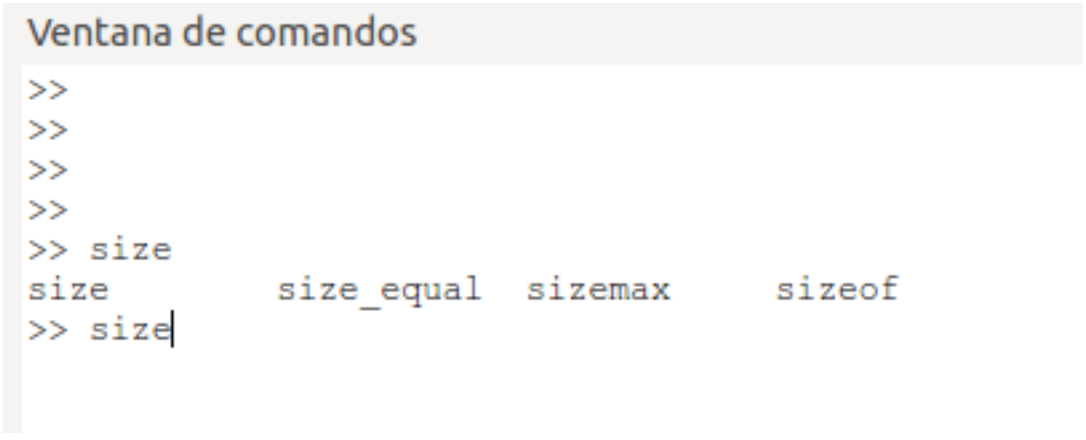
Una vez tecleados algunos comandos en la consola los podremos recuperar pulsando la tecla de cursor hacia arriba. Pulsando sucesivamente podremos recuperar el comando buscado y, o bien ejecutarlo pulsando *return*, o bien editarlo y modificarlo antes de una nueva ejecución.

Esta opción es muy útil para volver a dar valor a variables matriciales y cosas así, o simplemente para agilizar la ejecución de llamadas a funciones realizadas con anterioridad.

2.3.2 Ayuda a teclear comandos

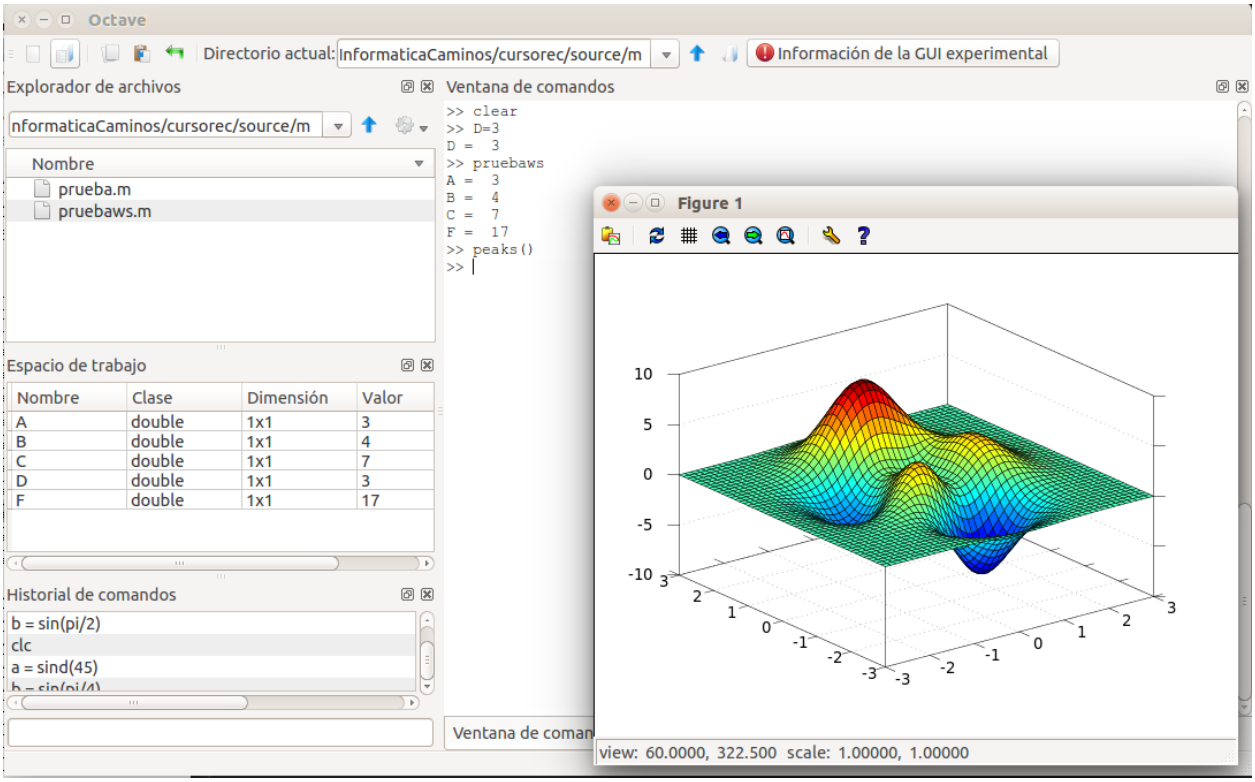
Cuando tecleamos comandos en la consola de Octave podemos utilizar la tecla del tabulador para que termine de completarse un comando parcialmente tecleado. Si solo hay una opción posible, el comando se completará. Si hay más de una opción, al teclear el tabulador por segunda vez nos mostrará las distintas opciones disponibles.

En el ejemplo que se muestra en la figura se tecleó '*size*' y dos veces el tabulador, obteniendo las distintas funciones disponibles que empiezan por '*size*':

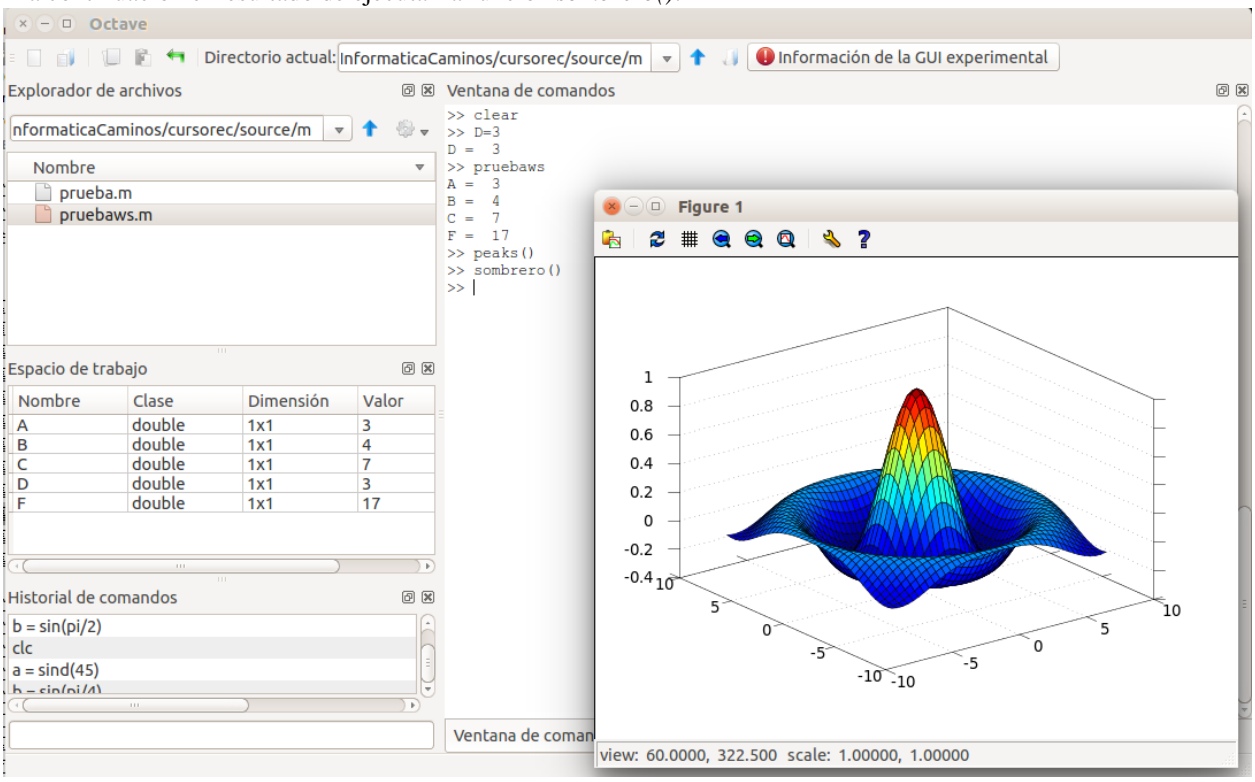


2.3.3 Comprobación de gráficos

Una vez instalado Octave conviene comprobar que funciona adecuadamente el sistema de gráficos. Octave ofrece dos funciones utilitarias para ese fin: *peaks()* y *sombrero()*. Prueba a utilizar ambas funciones. Como resultado se debe abrir una nueva ventana, la **ventana de gráficos**, mostrando la gráfica de la función. El resultado de la función *peaks()* debería ser parecido al de la figura siguiente:



Y a continuación el resultado de ejecutar la función *sombrero()*:



La ventana de gráficos puede variar un poco de unos sistemas operativos a otros. En todos los casos se ofrece un menú con opciones para añadir una rejilla, hacer zoom (prueba a pinchar y arrastrar sobre la ventana), para exportar el grafico a un fichero y otras.

Note: Algunos sistemas dan problemas al arrancar los gráficos. Si es tu caso, coméntalo en el foro y trataremos de resolverlo.

CHAPTER 3

Semana 2

3.1 Teoría de programación: Unidades de medida y orden de magnitud

Vamos a comenzar el análisis del artículo de Dewdney, y lo vamos a hacer empezando por las 'estampas'. Es curioso ver los anuncios comerciales que aparecen. Se nota el paso del tiempo. Vamos a analizar el anuncio que sale en la cuarta página referente un ordenador IBM que se vendía por aquellas fechas. El artículo es de diciembre de 1984 y el ordenador que se presenta es uno de los pocos ordenadores personales que se podía comprar.

Pero antes de analizar las características de ese ordenador, vamos a repasar un poco las unidades de medida que se utilizan en informática para medir la información. La unidad básica de medida de la información es el **bit**. Un *bit* es una unidad de información que puede tomar dos valores: **verdadero** o **falso**, que se suelen representar por **1** y **0**, respectivamente. Los ordenadores utilizan grupos de bits durante el procesamiento. Por otra parte, los documentos y programas que guardamos en los dispositivos de almacenamiento están también codificados en agrupaciones de bits para definir los caracteres de que constan. Se definen unas unidades múltiplo del bit para poder medir la cantidad de información. El **byte**, que equivale a ocho *bits*, es la unidad en la que se suelen basar las medidas.

- **byte** : Un byte son ocho bits. Si lo interpretamos como un número en base binaria puede almacenar un número entero entre 0 y 255. Si queremos considerarlo como un entero que pueda ser positivo o negativo, puede

representar un número entre -128 y 127. También podemos representar con un *byte*, un caracter de texto simple (del alfabeto inglés, sin caracteres especiales, ñ, acentuadas, ...).

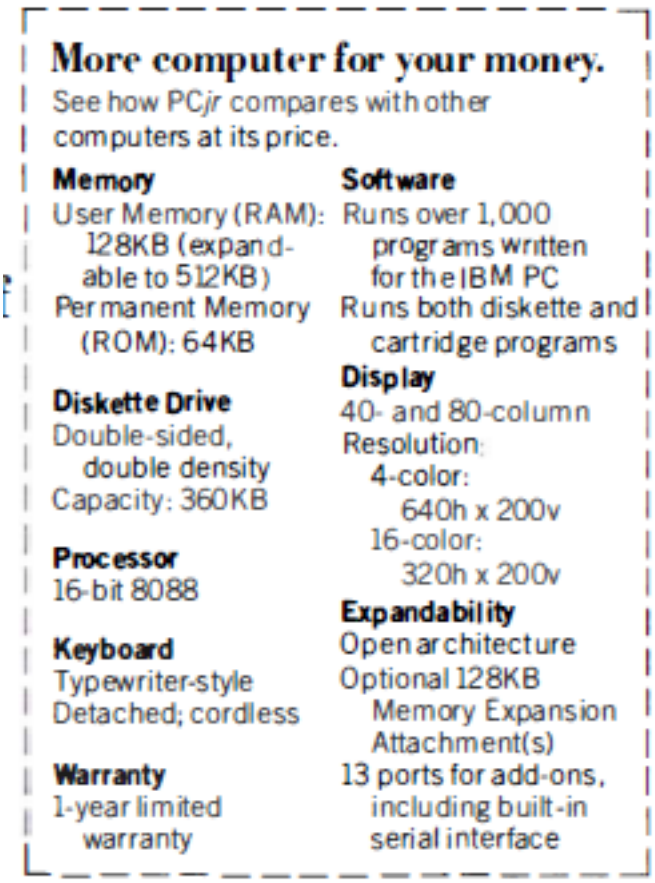
Note: El tipo de datos **int8** de Octave se corresponde con un número entero de un byte de tamaño, pudiendo guardar números enteros entre -128 y 127. Puedes comprobarlo en la consola de Octave tecleando **intmin('int8')** y **intmax('int8')**

- **dos bytes**: Con dos bytes se pueden representar los caracteres y dígitos de los distintos alfabetos existentes, incluyendo todo tipo de caracteres especiales, signos de puntuación, etc.

Note: En Octave existe un tipo de datos denominado **int16** que utiliza dos bytes para el almacenamiento y representa números enteros entre -32768 y 32767. Puedes comprobarlo en la consola de Octave tecleando **intmin('int16')** y **intmax('int16')**

- **kilobyte, kB**: Un kilobyte se utiliza como el equivalente de 1024 bytes (2^10) o de 1000 bytes, según el contexto. A efectos de orden de magnitud nos da igual uno que otro.
- **4 kilobytes**: El contenido de una página escrita a máquina, equivalente aproximadamente a 25 líneas de 80 caracteres cada una, incluyendo espacios.
- **Megabyte**: Un *megabyte*, o un **mega** como se suele abreviar, equivale a un millón de bytes. Equivaldría a 250 páginas escritas.
- **Gigabyte**: Un *gigabyte*, o un **giga** como se suele abreviar, equivale a mil millones de bytes, o lo que es lo mismo, mil libros de 250 páginas cada uno.
- **Terabyte**: Un *terabyte*, o **tera** como se suele abreviar, equivale a mil gigas, un millón de millones de bytes, 10^12 bytes, un millón de libros de 250 páginas cada uno.

Volvamos al anuncio del *IBM PC*. En el cuadro resumen de características podemos leer lo siguiente:



More computer for your money.

See how PC*jr* compares with other computers at its price.

Memory
User Memory (RAM): 128KB (expandable to 512KB)
Permanent Memory (ROM): 64KB

Diskette Drive
Double-sided, double density
Capacity: 360KB

Processor
16-bit 8088

Keyboard
Typewriter-style
Detached; cordless

Warranty
1-year limited warranty

Software
Runs over 1,000 programs written for the IBM PC
Runs both diskette and cartridge programs

Display
40- and 80-column
Resolution:
4-color:
640h x 200v
16-color:
320h x 200v

Expandability
Open architecture
Optional 128KB Memory Expansion Attachment(s)
13 ports for add-ons, including built-in serial interface

- **Memoria RAM:** La memoria RAM es la que está disponible para cargar los programas y datos que ejecuta el ordenador, uno de los cuales es el sistema operativo. Marca la cantidad de información que podemos tener simultaneamente cargada para procesamiento. Es el sitio de donde el procesador coge la información y las instrucciones a ajecutar, y donde deposita los resultados. Cuando se apaga el ordenador, el contenido de la memoria RAM se pierde. El *IBM PC* tenía una memoria RAM de 128 kilobytes, expandible a 512 kB, o sea, el equivalente a 32 folios escritos expandible a 128 folios escritos. El ordenador sobre el que estoy escribiendo estas líneas es un ordenador portátil con 8 gigas de RAM, o sea, el equivalente a ocho mil libros de 250 páginas cada uno, 15625 veces más que la máxima expansión del *IBM PC*.
- **Memoria ROM:** Es una memoria cuyo contenido no se pierde aunque apaguemos el ordenador. Contiene los programas y datos básicos para cargar y poner en funcionamiento el sistema operativo. El *IBM PC* tenía 64 kB de memoria ROM, o el equivalente a 16 folios escritos. El ordenador sobre el que escribo tiene 2560 kB de memoria ROM, 640 folios escritos, 40 veces más que el *IBM PC*.
- **Diskette:** Para almacenar de manera definitiva datos y programas se utilizaban los denominados *diskettes*, que era un disco magnético de 5.25 pulgadas de diámetro. Tenían una capacidad, en aquel momento de 360 kB, utilizando las dos caras del disco. En cada disco se podía guardar el eqivalente a 90 folios escritos. Los discos actuales, denominados *DVD*, tienen capacidades de 4 gigas, o sea, unas once mil veces más.
- **Disco duro:** El ordenador del anuncio no tenía disco duro. Sí que había modelos parecidos con disco duro. Empezaron siendo de 10 megabytes y enseguida pasaron a 20 megabytes. El equivalente a cinco mil folios escritos. El disco duro del ordenador que estoy utilizando tiene 1 **terabyte**, un millón de libros de doscientas cincuenta páginas, cincuenta mil veces más que los discos duros de entonces.
- **Pantalla:** Las pantallas podían trabajar en modo caracteres, similar a la consola de Octave, o en modo gráfico. En el caso de trabajar en modo caracteres se podía utilizar una pantalla 40 o de 80 caracteres de ancho. En el modo gráfico se podían utilizar gráficos de 320x200 puntos, con 16 colores posibles para los puntos, o con

gráficos de 640x200 puntos, con un máximo de 4 colores posibles. Las pantallas actuales para ordenadores funcionan siempre en modo gráfico, con resoluciones de 1920x1080 puntos y con 16 millones de colores posibles para cada ppunto.

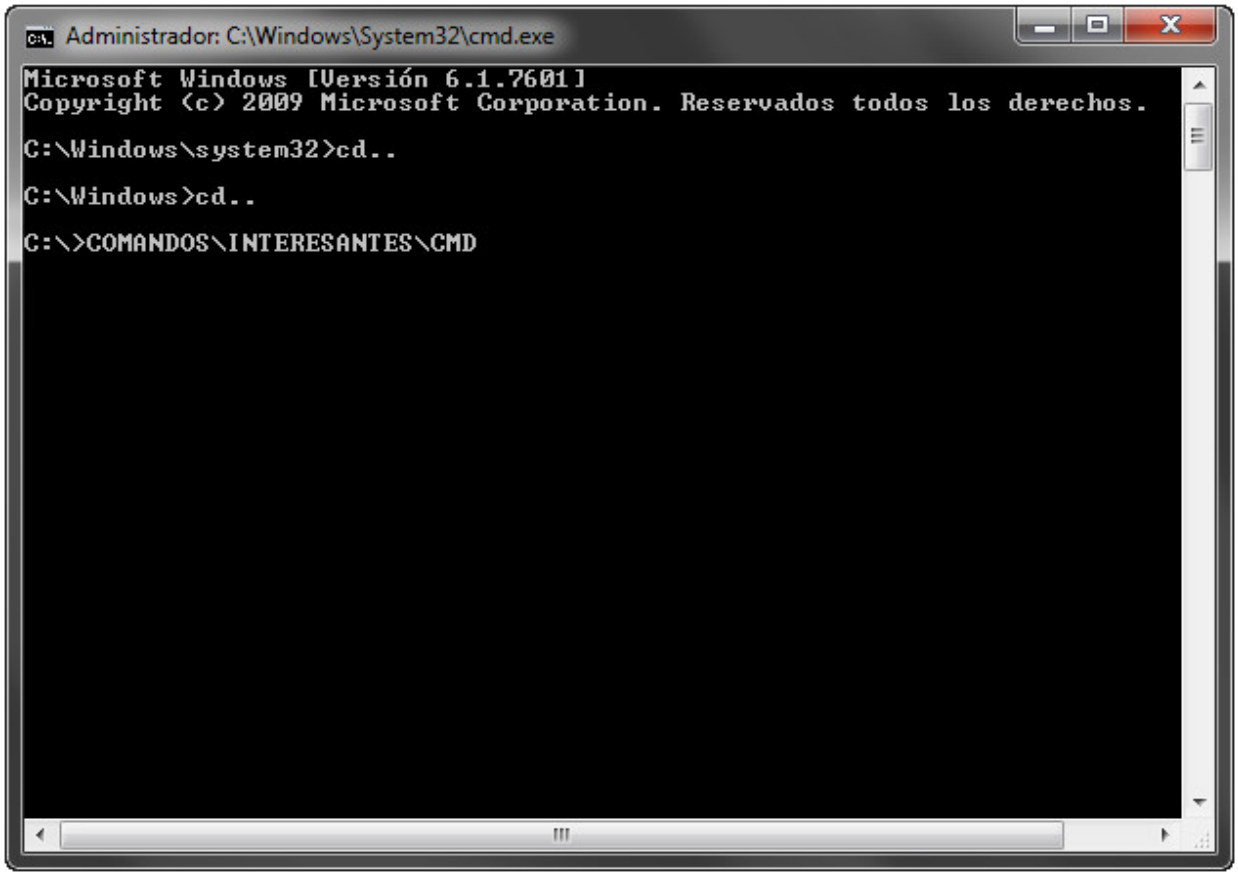
Note: Cada punto iluminado de la pantalla se denomina **pixel**. La *resolución* de una pantalla es el número de puntos que puede dibujar. Hay otro parámetro que marca la calidad y legibilidad de una pantalla, la **densidad de pixels**, que se mide en una unidad denominada **dpi**, *dots per inch*, puntos por pulgada. La unidad de densidad *dpi* es el número de pixels que hay en dos centímetros y medio de ancho de pantalla. No se ve igual una imagen de 100x100 puntos en una pantalla que tenga 10 centímetros de ancho que en una que tenga 20 centímetros de ancho. Los móviles tienen resoluciones de 800x600 pixels, por ejemplo, con una densidad alta, por ejemplo 160 puntos por pulgada, lo que favorece su legibilidad.

- **Procesador:** El procesador va recibiendo datos a través del llamado *bus* a intervalos marcados por un reloj de frecuencia determinada. El anuncio habla del procesador Intel 8088 de 16 bits, el reloj seguramente tendría 4.77 MHz. En cada impulso del reloj entraban 16 bits al procesador (ver nota). Podríamos establecer un parámetro bits/segundo, resultando que el Intel 8088 procesaba 4.77e6*16= 76.32e6 b/s. El ordenador en el que estoy trabajando tiene un procesador Intel de 64 bits a 2.50GHz. Si calculamos el mismo parámetro tendríamos 2500e6*64=160000e6 b/s, o sea, la capacidad de procesamiento de mi ordenador, medida con este parámetro, es más de dos mil veces la del IBM PC.

Note: En realidad el bus externo del procesador Intel 8088 era de ocho bits, eran los registros del procesador los que eran de 16 bits. (Fuente Wikipedia)

3.2 Tips del sistema operativo 2: la consola y el path

3.2.1 La consola del sistema operativo



Todos los sitemas operativos nos ofrecen una consola para acceder a las operaciones del mismo. El aspecto es similar en todos: una consola de texto, con un prompt esperando la entrada de un comando. Las respuestas a los comandos tecleados se muestran a través de la misma consola.

Pensar en las consolas como la forma antigua de utilizar el ordenador, por contraposición a la utilización a través de los interfaces gráficos basados en ventanas y utilización del ratón es un error. La consola es un método más eficiente de realizar determinadas tareas. En algunos sistemas incluso es la única manera de acceder al computador. Lo que sucede es que la curva de aprendizaje de la consola es mayor, y su uso se suele limitar a los usuarios medios y avanzados.

Los programadores deben realizar un uso intensivo de los sistemas de archivos, y es conveniente que conozcan al menos algunos comandos básicos de consola.

Los usuarios de Linux disponen de consolas muy potentes que permiten hacer cualquier cosa que se pueda hacer con el ordenador.

En Windows 7 podemos accedera la consola mediante **Inicio> Todos los programas> Accesorios> símbolo del sistema**. También mediante la barra de búsqueda o archivos del menú de inicio introduciendo "**cmd**" o "**símbolo del sistema**". Pulsando en ellos con el botón derecho del ratón podemos elegir el acceso en modo usuario o administrador. En Windows 8, el acceso más sencillo lo encontramos en el menú de usuario haciendo clic secundario en el botón de inicio.

Puedes echar un vistazo al siguiente artículo que habla de cómo acceder a la consola y cómo utilizar algunos de sus comandos en un sistema operativo Windows.

Comandos CMD, la consola de Windows también existe

3.2.2 El path del sistema

PATH es una variable de entorno de los sistemas operativos. El contenido de la variable *path* es una lista de directorios. Estos directorios son las rutas en las cuales el intérprete de comandos debe buscar los programas a ejecutar. Cada vez que ordenamos ejecutar un comando, el sistema operativo buscará el programa correspondiente en el directorio actual y en la lista de directorios que hay guardados en la variable *path*.

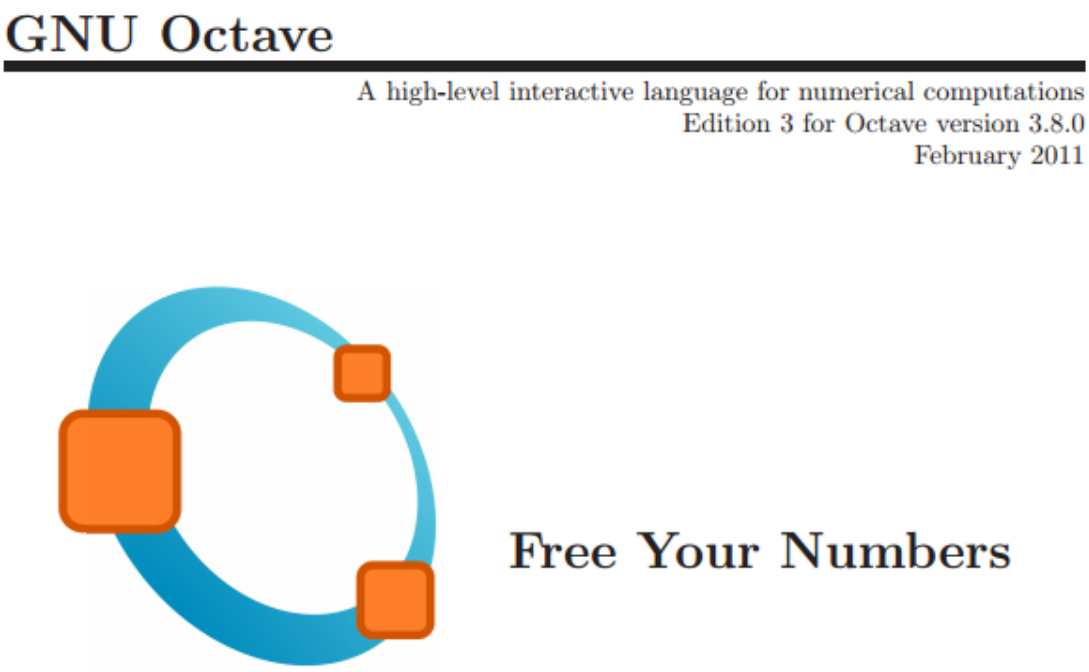
Para que funcionen bien algunos programas es necesario incluir su directorio en la variable *path*. Es el caso de Octave.

En Windows puedes ver el contenido de la variable *path* abriendo una consola y tecleando el comando *path*. También puedes ver o modificar el contenido a través de herramientas gráficas. Echa un vistazo al siguiente artículo, donde se explica cómo hacerlo.

How to set the path in Windows 7

Note: El interprete de Octave también tiene su propia variable *path*, que es una lista de directorios donde buscará los ficheros *.m* que ordenamos ejecutar desde la consola. Podemos ver el contenido de la variable *path* del interprete de Octave tecleando *path* en la consola. También podemos modificar la variable *path*. Para conocer más acerca del uso de la función *path()* teclea en consola *help path*.

3.3 Tips Octave 2: Octave en las redes sociales



La presencia de Octave en las redes sociales permite interactuar a la comunidad de usuarios de Octave, compartiendo experiencias, recursos y soluciones de problemas comunes. Repasamos a continuación algunos de los perfiles sociales de Octave en la Web:

- **Portal de Octave:** El primer sitio de referencia para Octave es el portal oficial de Octave en [1]. En este portal es donde tendremos a disposición las últimas versiones estables de Octave para los distintos sitemas operativos.

Es el lugar donde está la documentación oficial de Octave. Podemos acceder a la documentación oficial a través de la red, en formato html, en [2] o en formato pdf para descargar en [3].

- **Lista de correo:** Existe una lista de correo en la que los usuarios de Octave plantean las dudas que les surgen y otros usuarios, o los propios desarrolladores ayudan a resolverlas. También se anuncian en la lista todas las novedades acerca de nuevas versiones, actualizaciones de los paquetes, etc. Cualquiera puede suscribirse a la lista de correo en [4]. El histórico de mensajes de la lista se puede consultar en [5]
- **Twitter:** La cuenta de twitter **@GNUOctave**, no oficial, es una cuenta que tuitea acerca de novedades y recursos relacionados con Octave. No tiene mucho movimiento, pero es interesante mirar el histórico de tuits, pues tiene una buena colección de trucos, portales interesantes y demás. Puedes acceder al perfil de *@GNUOctave* en [6].
- **Facebook:** También hay una página en *Facebook* dedicada a Octave. No publican posts, pero tiene una buena lista de recursos relacionados con Octave. La página se puede consultar en [7]
- **Linkedin:** También encontramos varias páginas en Linkedin dedicadas a usuarios de Octave. Un ejemplo lo puedes ver en [8]
- **Google+:** En Google+ podemos acceder al perfil de GNU OCtave en [9]

[1] Portal de Octave

[2] Documentación WEB Octave

[3] Documentación de Octave en pdf

[4] Lista de correo de usuarios de OCtave

[5] Histórico de mensajes de la lista de correo de Octave

[6] Perfil en Twitter de @GNUOctave

[7] Página Facebook de GNU OCtave

[8] Linkedin Octave users and developers

[9] Google+ GNU OCtave

CHAPTER 4

Semana 3

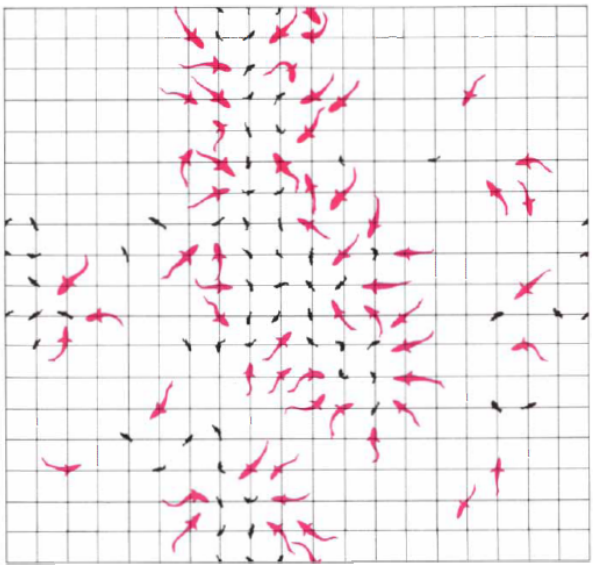
4.1 Tiburones y peces: un primer vistazo

Vamos a echar un primer vistazo al contenido del artículo de Dewdney. Se trata del artículo titulado '*Sharks and fish wage an ecological war on the toroidal planet Wa-Tor*', de A.K. Dewdney, que apareció publicado en la sección *Computer recreations* de la revista *Scientific American* en Octubre de 1984 [1].

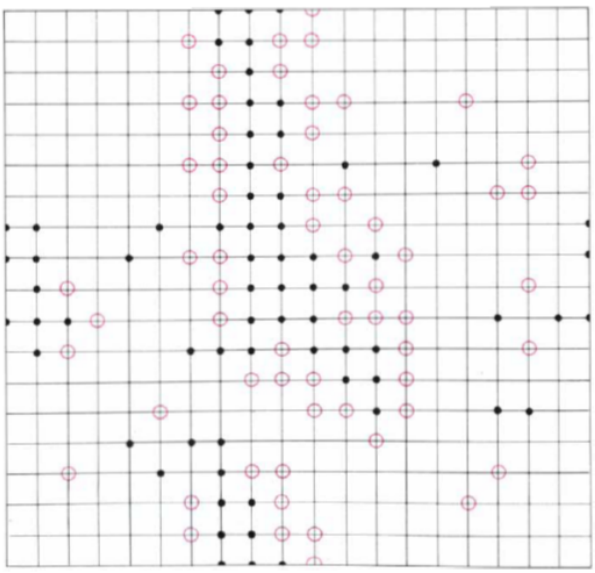
El artículo comienza hablando de un planeta imaginario llamado Wa-Tor, con forma toroidal o de donuts y totalmente cubierto de agua. Dice que hay dos especies viviendo en ese océano: tiburones y peces. Lo tiburones se alimentan de peces y los peces tienen siempre comida a disposición. Se trata de realizar un programa que simule los ciclos de alimentación y reproducción de los tiburones y peces y ver como evolucionan las poblaciones.

El programa, al que denominaremos WATOR, plantea una serie de reglas sencillas que gobiernan el comportamiento de los tiburones y los peces.

Para simular el océano toroidal, se utilizará una rejilla rectangular en la que cada borde esté *lógicamente* conectado con su borde opuesto. Para simular esto, cuando uno de los animales, por ejemplo, se desplace hacia el norte desde una de las casillas de la primera fila, el movimiento le llevará a la casilla situada en la misma columna, pero en la última fila. De la misma forma, un animal que se desplace hacia el este desde una casilla situada en la última columna, aparecerá en la casilla situada en la primera columna de la misma fila. Comportamientos similares se usarán para los movimientos hacia el sur desde la última fila o hacia el oeste desde la primera columna.



A realistic view of sharks eating fish



A more easily programmed view, in which circles represent sharks and dots represent fish

El artículo menciona dos versiones diferentes del programa realizadas por el autor y uno de sus ayudantes. En un caso se utilizó una rejilla de 80 puntos horizontales por 23 puntos verticales, y en el otro caso se utilizaron 23 posiciones en horizontal por 14 posiciones en vertical. Con un ordenador actual es posible realizar simulaciones en *océanos* mucho mayores sin problemas.

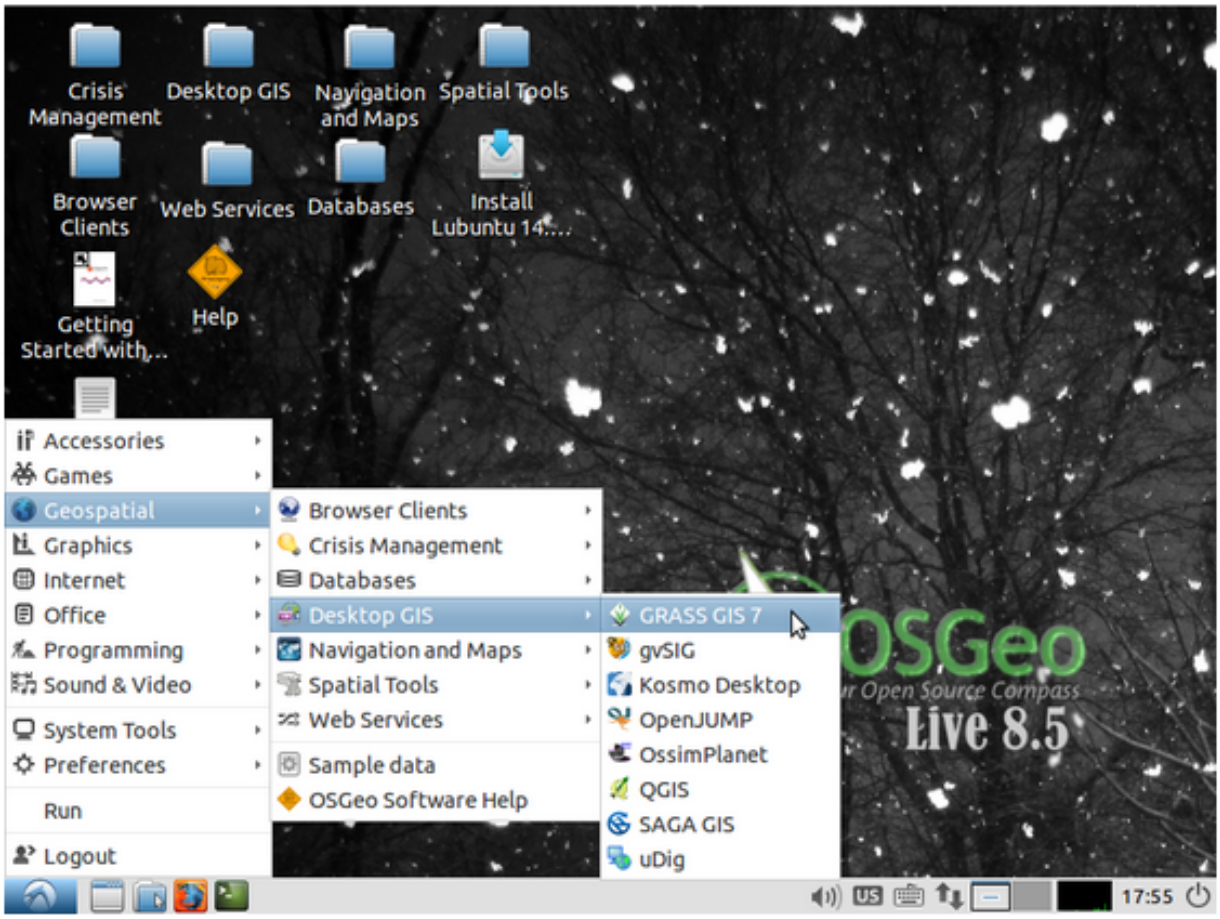
Se supone que el tiempo transcurre a intervalos discretos, que el autor denomina **chronos**. En cada *chrono*, WATOR mueve cada tiburón y cada pez. El desplazamiento será a una de las cuatro casillas adyacentes: norte, este, sur y oeste. La casilla se seleccionará al azar. En el caso de los peces, la casilla deberá estar libre, si no hay ninguna libre, el pez no se mueve en ese turno. En el caso de los tiburones el movimiento es un poco más complicado. El programa WATOR seleccionará al azar una de las casillas adyacentes que estén ocupadas por peces. El tiburón se desplazará a ella y devorará al pez. Si no hay casillas ocupadas por peces, seleccionará al azar una de las casillas adyacentes que estén desocupadas de otros tiburones.

El programa WATOR tiene cinco parámetros iniciales que se fijan al principio de cada simulación, y que en la nomenclatura original eran:

- **nfish:** número inicial de peces en el planeta. El programa WATOR distribuirá en el planeta, de una manera aleatoria más o menos uniforme, el número inicial de peces.
- **nsharks:** número inicial de tiburones en el planeta. El programa WATOR distribuirá en el planeta, de una manera aleatoria más o menos uniforme, el número inicial de tiburones.
- **fbreed:** número de chronos que deben transcurrir para que un pez tenga descendencia. Los peces dan lugar a la aparición de un nuevo miembro de la especie, al sobrevivir al número de ciclos indicado por *fbreed*.
- **sbreed:** número de chronos que deben transcurrir para que un tiburón tenga descendencia. Los tiburones dan lugar a la aparición de un nuevo miembro de la especie, al sobrevivir al número de ciclos indicado por *sbreed*.
- **starve:** número máximo de chronos que un tiburón puede estar sin comer. Si un tiburón está un numero de crhonos superior a *starve* sin comer, el tiburón muere y desaparece. Los peces se supone que tienen siempre comida.

Hasta aquí el planteamiento del problema. El resto del artículo se dedica a explicar como resolverlo. En las próximas semanas también nosotros iremos desgranando la solución del problema. Entre tanto, podéis ir meditando a cerca de cómo se podrían plantear las distintas partes del problema para resolverlo con Octave, que es nuestro objetivo. También podéis echar un vistazo a cómo lo resuelve Dewdney, leyendo el resto del artículo.

4.2 Tips del sistema operativo 3: Los Live DVD



Una distribución *live* o *Live CD* o *Live DVD* es un sistema operativo almacenado en un medio extraíble, tradicionalmente un CD o un DVD (de ahí sus nombres), que puede ejecutarse directamente en una computadora.

Normalmente, un Live CD viene acompañado de un conjunto de aplicaciones. Algunos Live CD incluyen una herramienta que permite instalarlos en el disco duro. Otra característica es que por lo general no se efectúan cambios en el ordenador utilizado.

Para usar un Live CD es necesario obtener uno (muchos de ellos distribuyen libremente una imagen ISO que puede bajarse de Internet y grabarse en disco) y configurar la computadora para que arranque desde la unidad lectora, reiniciando luego la computadora con el disco en la lectora, con lo que el Live CD se iniciará automáticamente (*Fuente*: Wikipedia).

Note: Una imagen ISO es un archivo donde se almacena una copia o imagen exacta de un sistema de ficheros, normalmente un disco compacto, un disco óptico, como un CD, un DVD, pero también soportes USB. Se rige por el estándar ISO 9660 que le da nombre. Para utilizar un archivo ISO, normalmente se monta la imagen en alguna unidad de CD/DVD virtual, o en su defecto se graba en un CD o DVD físico, aunque si lo que se quiere es evitarse complicaciones y ahorrar tiempo, basta con descomprimirlo con el programa WinRar o 7zip el cual se puede descargar de forma gratuita en la web de su autor, los cuales son capaces de descomprimir los archivos ISO sin problema. (*Fuente*: Wikipedia)

Vamos a presentar aquí dos distribuciones en Live DVD que son interesantes de conocer. La primera de ellas es el sistema operativo **Linux Ubuntu**. Podemos descargar un Live DVD en [1] con la última versión del sistema operativo

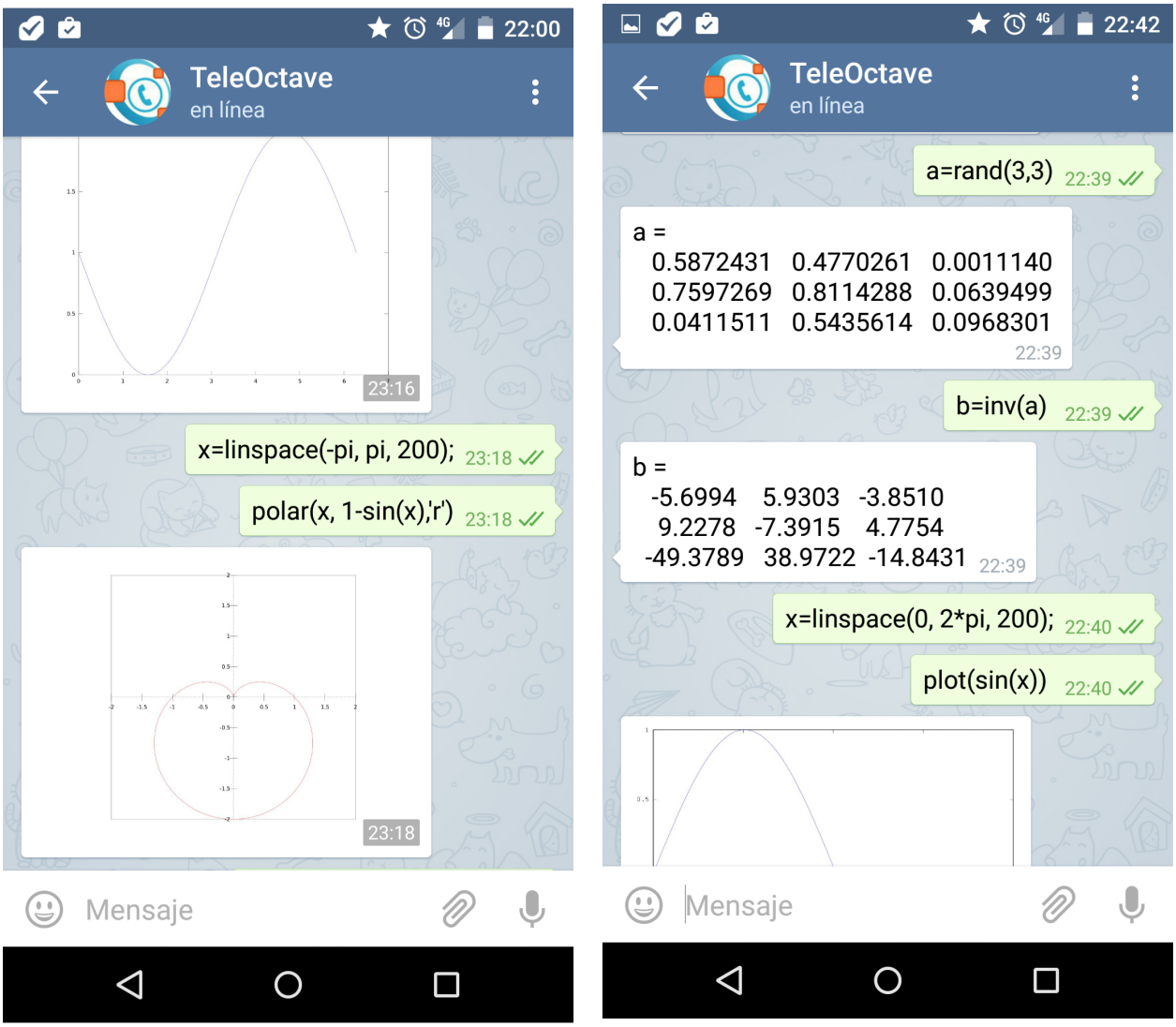
Linux Ubuntu, que es la distribución Linux más utilizada en ordenadores personales. Una vez montada la imagen ISO en un DVD o en un USB, podremos arrancar el ordenador y disponer de un sistema Ubuntu totalmente operativo y sin tocar nada de nuestro disco duro. Cuando apaguemos y volvamos a enecender, nuestro ordenador seguirá exactamente igual que antes de ejecutar el Live DVD. Esta opción nos permite probar Linux, antes de decidirnos por su instalación definitiva en el disco duro, o, simplemente, disponer de un Linux a mano por si necesitamos ejecutar algún programa del que no dispongamos versión para nuestro sistema operativo habitual.

El segundo Live DVD que os queremos presentar es el **Osgeo Live**. Se trata de un Live DVD con decenas de aplicaciones GIS instaladas y listas para utilizar. El disco lo confecciona y distribuye *OSGeo*, fundación sin animo de lucro dedicada al desarrollo y mantenimiento de programas de software libre para Sistemas de Información Geográfica y geomática. Entre los programas instalados están paquetes GIS de escritorio com gvSIG, QGIS, GRASS, bases de datos espaciales, como PostGIS, servidores de mapas como Geoserver, MapServer o MapProxi y muchos más programas GIS, perfectamente instalados y configurados y que se pueden utilizar como forma de aprendizaje o simplemente como forma de conocerlos mejor antes de decidirse por la instalación definitiva. Cada programa dispone de un *Quick View*, con instrucciones sencillas de como arrancarlos y dar los primeros pasos en los mismos. Podéis descargar el OSGeo Live y ampliar información sobre el mismo en [2].

[1] http://www.ubuntu.com/download/desktop/try-ubuntu-before-you-install

[2] http://live.osgeo.org/es/

4.3 Tips Octave 3: TeleOctave



El tip de Octave de esta semana está dedicado a **TeleOctave** [1], una forma curiosa de acercarse al mundo de Octave. Se trata de poder utilizar Octave a través de un contacto específico de la red **Telegram** [2]. Para los que no la conozcáis, *Telegram* es una red de mensajería instantanea muy similar a *WhatsApp*. Tiene algunas características aún no implementadas en WhatsApp, como la posibilidad de transmitir archivos de gran tamaño, la posibilidad de acceder, no solo a través de dispositivos móviles, sino también atravéss del navegador Web o de una aplicación de escritorio. Telegram posibilita además los chats secretos y otras mejoras respecto de WhatsApp en cuanto a la privacidad. El problema está en los usuarios, nuestros contactos es más probable que estén en WhatsApp que en Telegram, que es menos conocida. De todas formas, Telegram proporciona una opción configurable de forma que, si un contacto nuestro de WhatsApp se da de alta en Telegram, se nos añadirá automáticamente como contacto en Telegram, y recibiremos una notificación al respecto.

Pues bien, uno de los contactos que podemos tener en Telegram es **TeleOctave**. Cuando enviamos como mensaje un comando válido de Octave a TeleOctave, nos responderá como si de una consola de Octave se tratara, ¡incluso los gráficos!.

Podéis probarlo sin problema en vuestros móviles u ordenadores, porque entre las diferencias entre WhatsApp y Telegram está que Telegram es totalmente gratuito.

[1] http://teleoctave.altervista.org/

[2] https://telegram.org/

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*