

Introducción a QtOctave.

Índice de contenido

1 ¿Qué es QtOctave?.....	1
2 Instalación.....	1
3 Usando QtOctave.....	2
3.1 El terminal.....	3
3.2 Navegando a un directorio.....	3
3.3 El menú Data.....	3
3.4 El menú Plot.....	4
4 Menús creados por el usuario.....	5
4.1 Creando el menú de usuario Transformada de Fourier.....	5
4.2 Los archivos ".menu".....	6
4.3 Añadiendo iconos a los menús.....	7
4.4 Sugerencias a la hora de hacer menus.....	7
5 El WidgetServer.....	7
5.1 Ejemplo de uso del Widgetserver desde Octave.....	7
5.2 Ejemplo comentado.....	9

1 ¿Qué es QtOctave?

QtOctave es un front-end para Octave basado en Qt 4. Octave es una aplicación de cálculo matemático muy similar a Matlab.

2 Instalación.

Para instalar QtOctave se necesitan tener instalados:

- Octave.
- Las bibliotecas de desarrollo de Qt4. Generalmente llamadas libqt4-dev.
- El compilador de C++, por supuesto.

Para proceder a la instalación hay dos vías:

1. Usar el programa configure. Para ello hay que teclear la siguiente secuencia de comandos:

```
./configure
make
make install
```

Puede suceder que haya conflictos con la instalación de Qt 3 ó que el programa de instalación no sea capaz de encontrar la instalación de Qt 4. En el caso de tener este problema hay que usar la opción `--qtdir` que le indica al programa de configuración dónde puede encontrar la instalación de Qt 4. Por ejemplo, si se ha instalado Qt 4 en el directorio `/usr/local/Trolltech/Qt-4.1.1/` sólo hay que pasar al programa de configuración las siguientes opciones:

```
./configure --qtdir=/usr/local/Trolltech/Qt-4.1.1/
make
make install
```

También se puede usar la opción `--prefix` para indicar otro punto de instalación.

2. La otra vía es similar a la anterior, pero usando la opción `--assistant`:

```
./configure --assistant
make
make install
```

Funciona de forma similar al caso anterior pero en el momento en el que se encuentren las Qt 4, se procede a mostrar un asistente de instalación.

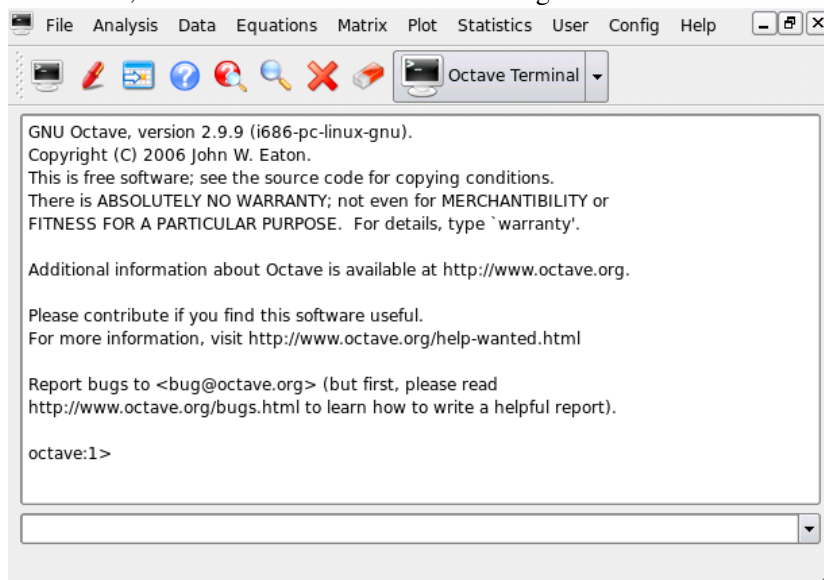
Otro posible problema es que en nuestra distribución de Linux favorita decida hayan cambiado el nombre a algunos programas de Qt 4 para no tener conflictos con Qt 3. Lo normal es que se cambie el nombre del comando qmake por qmake4 o algo similar. En este caso, se puede hacer un enlace simbólico a el comando qmake de forma que tenga el nombre correcto. También se puede usar la opción --qtdir para indicar el punto de instalación de Qt 4.

3 Usando QtOctave.









Una vez instalado QtOctave, para iniciarlo se teclea:

```
qt octave
```


Si todo funciona correctamente, se mostrará una ventana como la siguiente:



En esta ventana se pueden apreciar los siguientes iconos:

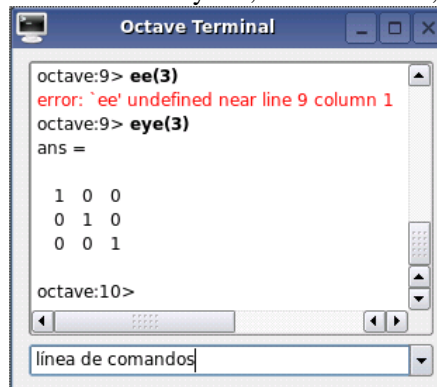
	Abre una nueva sesión de QtOctave.
	Abre un editor de texto. El editor se puede configurar en el menú Config.
	Abre una matriz en una hoja de cálculo.
	Abre la ayuda de Octave. Se puede configurar en el menú Config.
	Ayuda dinámica. Muestra ayuda de los comandos según se van tecleando en el terminal.
	Busca comandos similares al que se haya tecleado.
	Ctrl. + C. Sirve para parar procesos desbocados, bucles infinitos,...
	Borra los contenidos de el terminal.

Además, en la barra de iconos, aparece una lista desplegable con las ventanas abiertas.

En el caso de que no haya manejado nunca Octave, es recomendable usar la ayuda de Octave  y leer el apartado "A Brief Introduction to Octave".

3.1 El terminal.


En el terminal se puede ejecutar comandos de Octave y ver, en una ventana, la salida de dichos comandos:



```
Octave Terminal
octave:9> ee(3)
error: `ee` undefined near line 9 column 1
octave:9> eye(3)
ans =
  1 0 0
  0 1 0
  0 0 1
octave:10>
línea de comandos
```


En la línea de comandos se pueden ejecutar nuevos comandos y se puede ver una lista de los comandos anteriores. En el caso de que se produzca un error estos aparecerán en rojo.

Seleccionando un texto, en la salida, y haciendo click con el botón derecho del ratón, se pueden copiar fragmentos de texto de la salida.

A veces, el terminal se llena de "basura", por ello se puede usar el borrador , para borrar el terminal.

Puede ocurrir también que se entre en un bucle infinito. Por ejemplo, si se teclea:

```
while (2>1)
end
```


Se entrará en un bucle infinito. Con el botón , se pueden parar los procesos desbocados o los bucles infinitos, como el anterior.

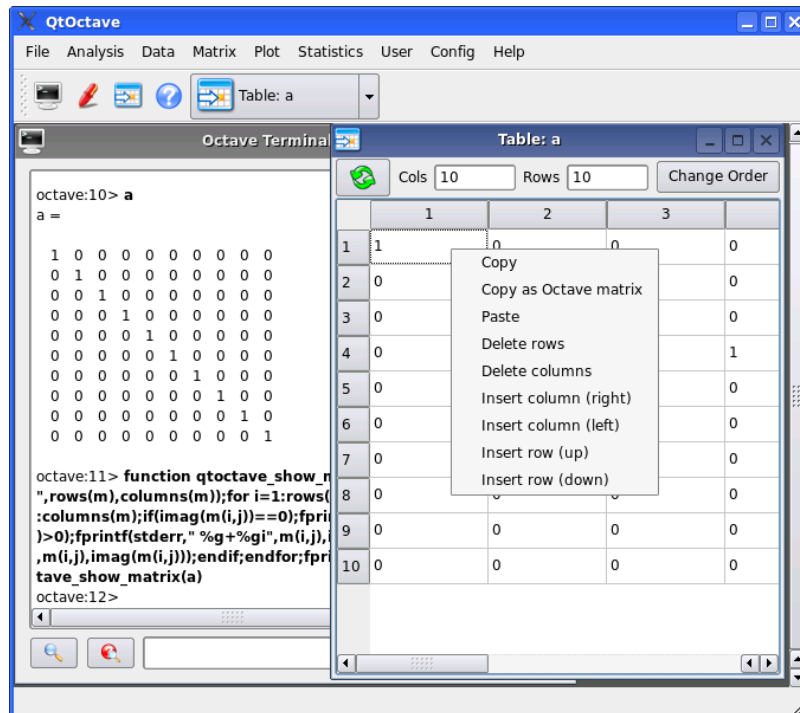
3.2 Navegando a un directorio.

Es habitual tener un directorio o carpeta con los ficheros de trabajo. Se puede usar la orden "cd" de Octave para navegar por los directorios. Pero es más práctico el menú "File/Change dir". Mostrará una ventana para seleccionar el directorio al que se quiera navegar y cambiará a dicho directorio.

3.3 El menú Data.

Este menú posee herramientas para realizar manipulación de datos.


Una herramienta interesante es la utilidad "Table" . Esta utilidad muestra matrices como si fueran hojas de cálculo. Primero preguntará el nombre de la matriz que se desea mostrar. Si todo es correcto, mostrará una ventana con la matriz. Como la de la figura siguiente:



Se pueden seleccionar celdas, filas o columnas. Si se hace click con el botón derecho del ratón aparecerá un menú con las diferentes posibilidades. Se puede copiar, pegar, borrar,...

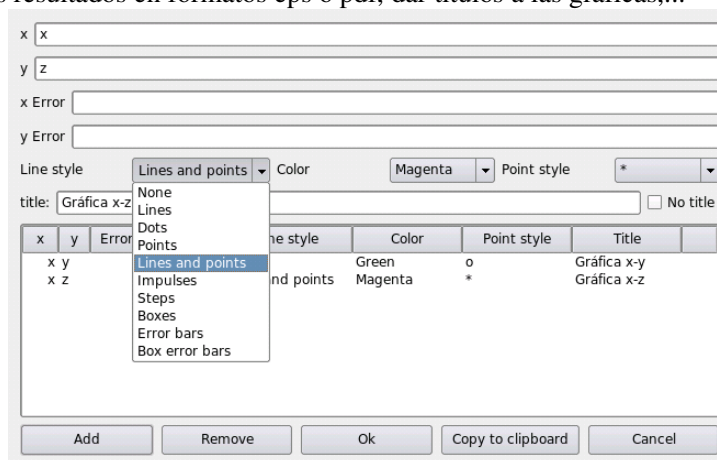
Una de las opciones más interesante es usar "Copy as Octave matrix". Esta opción permite copiar un trozo de matriz y después pegarla en un editor, usando la sintaxis de Octave para las matrices.

Las entradas "Cols" y "Rows", sirven para cambiar las filas y las columnas de la hoja de cálculo (el orden de la matriz).

A veces, se pueden realizar manipulaciones de matrices con Octave, que no se vean reflejadas en la hoja de cálculo. Usar el botón  para actualizar la hoja de cálculo.

3.4 El menú Plot.

Este menú tiene herramientas para trabajar con gráficas. Permite generar gráficas de una forma sencilla, usando un asistente, exportar los resultados en formatos eps ó pdf, dar títulos a las gráficas,...



Como en casi todos los asistentes de QtOctave, se pueden ejecutar los comandos necesarios en el terminal, o bien copiarlos al porta-papeles, para pegarlos en un editor. QtOctave no sólo pretende dar un entorno amigable para lidiar con Octave, sino que también se intenta facilitar la labor de la programación en Octave.

4 Menús creados por el usuario.

Los usuarios pueden crear sus propios menús de una forma sencilla. Estos menús se pueden colocar en dos lugares:

1. En "DIRECTORIO_DE_INSTALACIÓN/menus". Todo menú que se coloque aquí, será visible para el todos los usuarios de la máquina.
2. En "~/.qtoctave/menus/". Los menús que aquí se coloquen sólo serán visibles para el usuario.

Es una buena idea navegar por "DIRECTORIO_DE_INSTALACIÓN/menus", para ver la implementación de los menús usados por QtOctave.

Veamos un ejemplo de menú.

4.1 Creando el menú de usuario Transformada de Fourier.

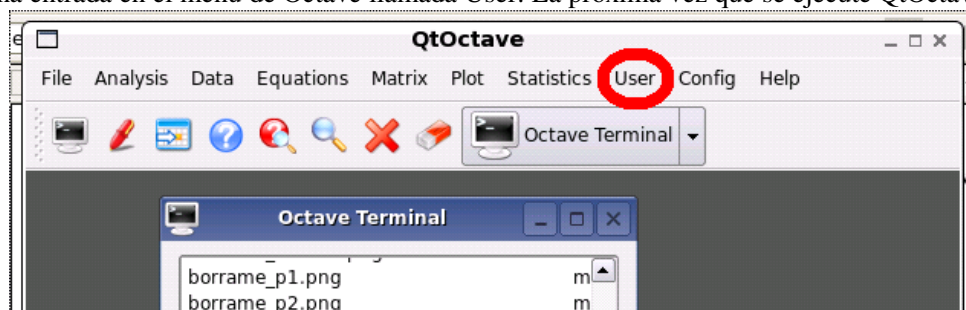
Supongamos que el usuario tiene que usar mucho la transformada de Fourier. Por ello le podría resultar útil colocarla en el menú.

Lo primero que debe de hacer este usuario es pensar dónde desea colocar dicha entrada. Si desea colocársela en un nuevo menú llamado "User", debe crear el directorio "User" en "~/qtoctave/menus/". Si desea que dicha entrada esté en el menú "Analysis", se debe crear un directorio llamado "Analysis" en "~/qtoctave/menus/".

Vamos a suponer que el usuario desea crear un nuevo menú "User". Deberá teclear en la consola (usando bash o sh):

```
mkdir -p ~/.qtoctave/menus/User
```

Esto creará una entrada en el menú de Octave llamada User. La próxima vez que se ejecute QtOctave, la mostrará:



Con nuestro navegador favorito o usando la consola, nos introducimos en dicho directorio:

```
cd ~/.qtoctave/menus/User
```

Usando nuestro editor favorito se crear un fichero llamado "fourier.menu" con los siguientes contenidos (en el siguiente apartado se verá lo que significan):

```
#FFT
menu_name=Fast Fourier Transform

#i1
input_label=Data (matrix)
input=

#i2
input_label=Total time
input=

#o1
```

```

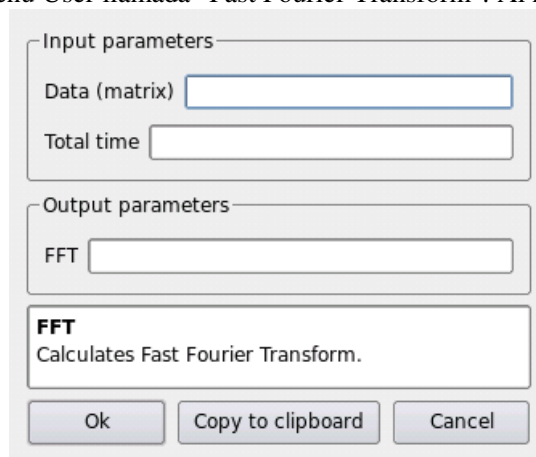
output_label=FFT
output=

begin{command}
%o1%=fft(%i1%);
plot( [0:length(%o1%)-1]*%i2%/length(%o1%) , abs(%o1%) );
end{command}

begin{help}
<b>FFT</b> <br>
Calculates Fast Fourier Transform.
end{help}

```

Esto genera una entrada en el menú User llamada "Fast Fourier Transform". Al hacer click sobre ella se muestra:



4.2 Los archivos ".menu".

En el apartado anterior se ha visto un ejemplo de fichero ".menu". Pasemos a comentar cada una de sus partes:
#FFT

Esto es un comentario y es ignorado por QtOctave. Toda línea que comience por # se considera un comentario.
menu_name=Fast Fourier Transform

Indica el nombre de menú. La entrada menu_name sirve para dar nombres a los menús.

```

#i1
input_label=Data (matrix)
input=

```

input_label sirve para crear una nueva entrada en la lista de parámetros de entrada. input sirve para indicar el nombre del parámetro por defecto. Por ejemplo, si se escribe input=x, el parámetro por defecto será x.

```

#i2
input_label=Total time
input=

```

```

#o1
output_label=FFT
output=

```

output_label sirve para crear una nueva entrada en la lista de parámetros de salida. output sirve para indicar el nombre del parámetro por defecto. Por ejemplo, si se escribe output=x, el parámetro por defecto será x.

```

begin{command}
%o1%=fft(%i1%);
plot( [0:length(%o1%)-1]*%i2%/length(%o1%) , abs(%o1%) );
end{command}

```

Cuando el usuario pulse el botón Ok, o bien, "Copy to clipboard", todo lo que esté entre begin{command} y end{command}, será enviado al terminal de Octave, o al portapapeles, respectivamente.

Hay que fijarse que %i1% es el nombre del primer parámetro de entrada, %i2% del segundo, ... En este caso %i1% será el nombre que teclee el usuario en "Data (matrix)", %i2% será el nombre que teclee el usuario en "Total time".

%o1% será el nombre del primer parámetro de salida, %o2% del segundo,...

En este caso %o1% será el nombre que el usuario teclee en "FFT".

```
begin{help}
<b>FFT</b> <br>
Calculates Fast Fourier Transform.
end{help}
```

Todo lo que esté entre begin{help} y end{help} se mostrará en un área de texto y será usada como ayuda del menú. Como se puede ver en el ejemplo, el texto entre begin{help} y end{help} deberá ser html. El html soportado es muy sencillo.

4.3 Añadiendo iconos a los menús.

Se pueden añadir iconos a los menús. En el lugar donde se haya colocado el fichero ".menu", hay que poner el icono no en mismo nombre con la extensión en ".png".

Por ejemplo, si se dispone del menú "about.menu", el icono se deberá poner en el mismo lugar y llamarse: "about.menu.png".

Todos los iconos deben estar en formato png.

4.4 Sugerencias a la hora de hacer menus.

Es bueno ver los ejemplos de menú que hay en "DIRECTORIO_DE_INSTALACIÓN/menus".

Para crear un menú dentro de otro, hay que crear un directorio dentro de otro en el lugar donde se esté el directorio que contiene a los menús.

Siempre se dispone del botón "Copy to clipboard". Hace que los comandos se copien al porta-papeles.

Los asistentes que se pueden crear con los archivos ".menu" son muy simples. Si necesita mostrar algo más complicado use el WidgetServer. Por favor, lea la documentación del WidgetServer.

5 El WidgetServer.

El WidgetServer es un programa para la creación de ventanas. Se pueden crear ventanas desde prácticamente cualquier lenguaje de programación, incluido Octave. Por favor, lea el tutorial del WidgetServer para ver cómo se crean las ventanas.

5.1 Ejemplo de uso del Widgetserver desde Octave.

Las comunicaciones con el Widgetserver se realizarán usando el comando de octave **popen2**. El problema es que la implementación de este comando falla. La implementación correcta de este comando sería:

```
function [in, out, pid] = popen2 (command, args)

in = -1;
out = -1;
pid = -1;

if (nargin == 1 || nargin == 2)
```

```

if (nargin == 1)
    args = "";
endif

if (ischar (command))

    [stdin_pipe, stdin_status] = pipe ();
    [stdout_pipe, stdout_status] = pipe ();

    if (stdin_status == 0 && stdout_status == 0)

        pid = fork ();

        if (pid == 0)

            ## In the child.

            fclose (nth (stdin_pipe, 2));
            fclose (nth (stdout_pipe, 1));

            dup2 (nth (stdin_pipe, 1), stdin);
            fclose (nth (stdin_pipe, 1));

            dup2 (nth (stdout_pipe, 2), stdout);
            fclose (nth (stdout_pipe, 2));

            if (exec (command, args) < 0)
                error ("popen2: unable to start process `%s'", command);
                exit (0);
            endif

        elseif (pid)

            ## In the parent.

            fclose (nth (stdin_pipe, 1));
            fclose (nth (stdout_pipe, 2));

            %if (fcntl (nth (stdout_pipe, 1), F_SETFL, O_NONBLOCK) < 0)
            % error ("popen2: error setting file mode");
            %else
            in = nth (stdin_pipe, 2);
            out = nth (stdout_pipe, 1);
            %endif

            elseif (pid < 0)
                error ("popen2: fork failed -- unable to create child process");
            endif
        else
            error ("popen2: pipe creation failed");
        endif
    else
        error ("popen2: file name must be a string");
    endif
else
    usage ("[in, out, pid] = popen2 (command, args)");
endif

endfunction

```

Se puede incluir dentro de los programas de Octave, pues Octave permite redefinir las funciones.

Para comunicarse con Octave se seguirán los siguientes pasos:

1. Se abrirán las comunicaciones con el Widgetserver usando `popen2`.
2. Se le mandarán las instrucciones necesarias para construir la ventana.
3. Se escucharán los eventos con `fgets`.

5.2 Ejemplo comentado.

El siguiente ejemplo abre una ventana que pide el argumento de la función sombrero:

```
[out,in,pid]=popen2("widgetserver");
if(pid<0)
    printf("Error widgetserver couldn't be executed\n");
    exit(1);
end
```

Lo que se ha hecho es abrir las comunicaciones con el Widgetserver. En el caso de que el programa `widgetserver` no se encuentre, se muestra un mensaje de error. En el caso de que no funcione correctamente, se puede usar la implementación de `popen2` mostrada anteriormente.

```
fprintf(out, "<window:frame>\n");
fprintf(out, " <title:Sombrero/>\n");
fprintf(out, " <p>\n");
fprintf(out, " <label:l1>\n");
fprintf(out, "         Number of grids\n");
fprintf(out, " </label>\n");
fprintf(out, " <lineedit:w0>\n");
fprintf(out, "         <text: 20/>\n");
fprintf(out, " </lineedit>\n");
fprintf(out, " </p>\n");
fprintf(out, " <html:ht1>\n");
fprintf(out, "         Draws sombrero function\n");
fprintf(out, " </html>\n");
fprintf(out, " <p>\n");
fprintf(out, " <button:ok>\n");
fprintf(out, "         <listen: clicked/>\n");
fprintf(out, "         <text>\n");
fprintf(out, "             Ok\n");
fprintf(out, "         </text>\n");
fprintf(out, " </button>\n");
fprintf(out, " <button:cancel>\n");
fprintf(out, "         <listen: clicked/>\n");
fprintf(out, "         <text>\n");
fprintf(out, "             Cancel\n");
fprintf(out, "         </text>\n");
fprintf(out, " </button>\n");
fprintf(out, " </p>\n");
fprintf(out, " </window>\n");
```

```
fflush(out);
```

Se envía la ventana que se tiene que mostrar. *El `fflush(out)` es importante para poder enviar los datos.* Es mucho más cómodo escribir la ventana en un fichero y enviarla después leyendo el fichero. Leer el tutorial del WidgetServer para entender la forma en que se crea la ventana.

```
%Event loop
while( isstr ( line=fgets(in) ) )
    % Se procesan los eventos para cerrar la ventana
    if(
        ( length(line)>=15 && strcmp(substr(line,1,16),'*clicked:
cancel') )
        ||
```

```

        ( length(line)>=13 && strcmp(substr(line,1,13),'*close: fra-
me') )
    )
    fprintf(out, "<quit/>\n");
    fflush(out);
    break;
end;

% Se debe haber hecho click sobre el botón de Ok.

% Se lee la línea de texto.
fprintf(out, "<lineedit:w0>\n");
fprintf(out, " <getText/>\n");
fprintf(out, "</lineedit>\n");
fflush(out);

line=fgets(in);
w0=fgets(in);

%Se dibuja el sombrero
sombrero(eval(w0));

end

```

Este era el bucle de eventos, se van leyendo, línea a línea los datos que envía el Widgetserver y se procesan. Las funciones strcmp, substr y length se usan para comparar las cadenas. La función strcmp compara dos cadenas e indica si son iguales. length sirve para indicar la longitud de un array (una cadena). substr sirve para extraer una parte de una cadena.

```

fclose(in);
fclose(out);

% Fin del programa

```

Si se ejecuta este programa desde Octave, se obtiene la siguiente salida:

