

Física Computacional

Capítulo 11: Visualización de datos con OpenDX

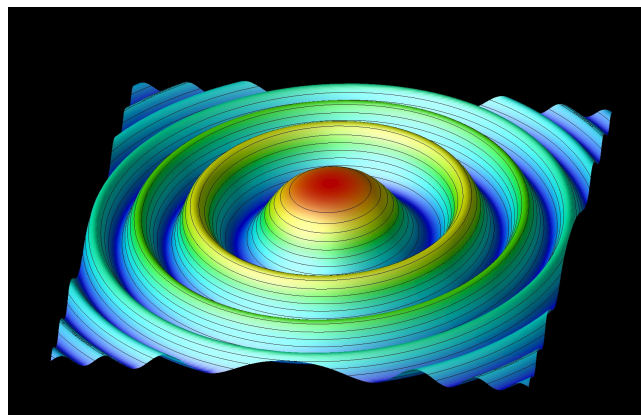
Autores:

Luis Molina e Iván Cabria

Universidad de Valladolid

Mayo 2014

En este capítulo se ofrece una pequeña introducción al uso del programa OpenDX. Este paquete, de muy sencilla instalación bajo Ubuntu, permite representar de forma elegante datos en 3 dimensiones. En el tutorial se enseñan los fundamentos de un entorno de programación visual que permite procesar y visualizar datos complejos mediante el empleo de una amplia variedad de herramientas. Finalmente, se ofrecen además algunos ejemplos útiles que pueden aplicarse de forma directa a la representación de datos físicos de interés (propagación de ondas, visualización de funciones de onda tridimensionales, etc...).



Índice

1. Instalación y lectura de datos.	2
2. Manejo básico del editor visual de programas	4
3. Datos tridimensionales. Isosuperficies	7
4. Otros ejemplos interesantes	9

Sección 1

Instalación y lectura de datos.

Para instalar el programa en Ubuntu, simplemente se debe abrir el Centro de Software de Ubuntu, buscar “Opendx”, e instalar el software. Se recomienda instalar también los complementos **dx-doc** (documentación) y **dxsamples** (ejemplos).

Como primer paso, antes de representar gráficamente la información de nuestro fichero de datos (datos.dat, por ejemplo), debemos generar un fichero auxiliar para OpenDX (datos.general) donde se especifica la forma en la que los datos se organizan dentro del fichero.

Tomaremos como ejemplo el programa `funcion-2d.c`, que evalúa los valores de la función $\cos^2(x^2 + y^2)e^{-(x^2+y^2)/\pi^2}$ en una malla regular en el plano xy . El programa trabaja utilizando dos variables: `xmin` (que controla el intervalo $[-xmin, xmin]$ en el que se representan los datos en cada dirección) y `Grid_size` (el número de datos tomados en cada dirección). La parte del programa que calcula e imprime en un fichero el conjunto bidimensional de datos es:

```
for(i=0;i<Grid_size;i++) {
    ui = (2.0*xmin)*((double)i-(double)Grid_size/2.0)/(double)Grid_size;
    for(j=0;j<Grid_size;j++) {
        uj = (2.0*xmin)*((double)j-(double)Grid_size/2.0)/(double)Grid_size;
        x = f(ui,uj) ;      fprintf(output,"% .10lf \n",x);
    }
}
```

A la hora de organizar los datos, vemos que se coloca un dato en cada línea. En nuestro caso, vemos que los datos se organizan de forma que la coordenada x es controlada por el bucle **externo**, mientras que la coordenada y es controlada por el bucle **interno**. Si quisiéramos representar datos tridimensionales, deberíamos añadir un tercer bucle dentro del bucle sobre la coordenada z , para calcular los valores de la coordenada z .

Comenzaremos por abrir el programa escribiendo en un terminal el comando:

dx &

Para generar el fichero auxiliar .general, debemos hacer click en la opción “**Import Data**” del menú principal (ver Figura 1).



Figura 1: Menú principal del programa OpenDX.

Se abre entonces una ventana llamada “**Data Prompter**”, donde debemos marcar la casilla “**Grid or Scattered file**”. Seguidamente, debemos clicar la opción “**Describe Data**” En la ventana que se abre a continuación (ver Figura 2), deberemos introducir los siguientes datos:

- El nombre del fichero de datos a utilizar (por ejemplo, `datos-funcion-2d.dat`).
- El número de puntos de la grid en cada dirección (200 x 200, en nuestro caso).
- En cada dirección, el valor mínimo de la coordenada x, y, z, etc..., seguido del tamaño del intervalo entre datos. En nuestro ejemplo (consultar de nuevo el programa `funcion-2d.c`), el valor mínimo de las coordenadas x e y es, en ambos casos, -3.141592. Debido a que los datos van desde $-\pi$ hasta $+\pi$, y hay 200 puntos, el valor del intervalo será igual a $2\pi/200 = 0.031415$.

En la ventana de lectura de datos existe una opción (**Data order**), con dos posibles valores (**row** ó **column**). Debemos dejar marcada la opción por defecto (**row**), que corresponde a la ordenación de datos utilizada en el programa `funcion-2d.c`, con el bucle sobre los datos y interior, y con el bucle sobre los datos x exterior.

Es importante entender que, con tal convención, el software OpenDX entenderá que la primera coordenada de datos corresponde a la coordenada x, la segunda a la coordenada y, etc... Por tanto, para casos en los que utilicemos rejillas rectangulares (200 x 100, ó 100 x 200) en lugar de cuadradas, se debe seguir la regla de que **la primera dimensión corresponde al bucle más externo del programa de generación de datos**.

Una vez introducidos todos los datos, se debe emplear la opción “**Save as...**” del menú del Data Prompter para guardar el archivo auxiliar de tipo .general. Hecho esto, podemos ya cerrar las ventanas y comenzar a escribir programas para OpenDX desde el “**Visual Program Editor**”.

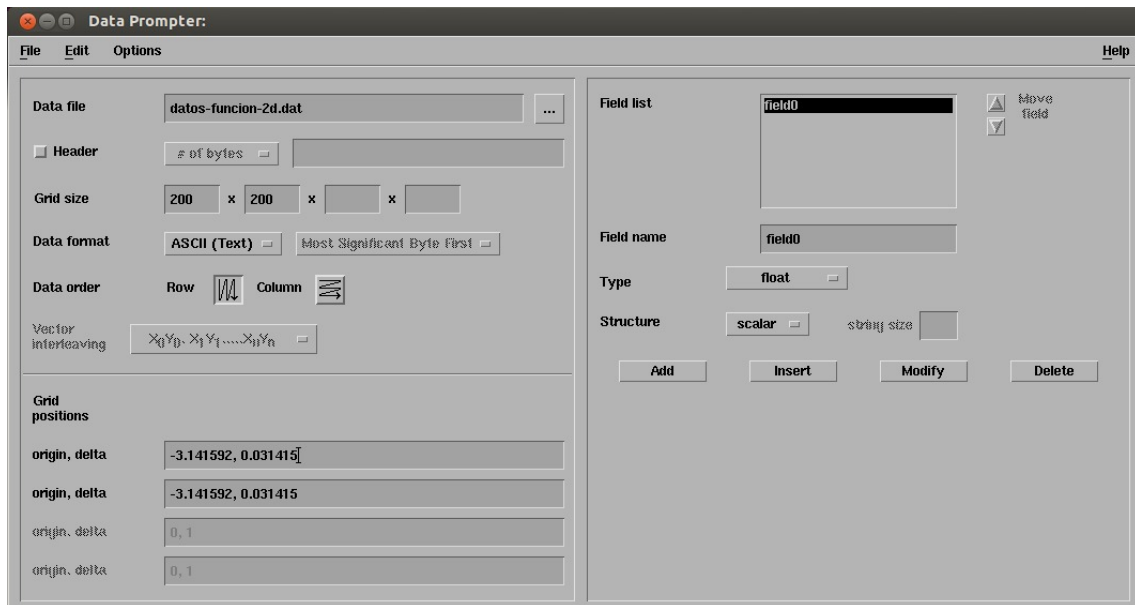


Figura 2: Ventana de lectura de datos en OpenDX.

Sección 2

Manejo básico del editor visual de programas

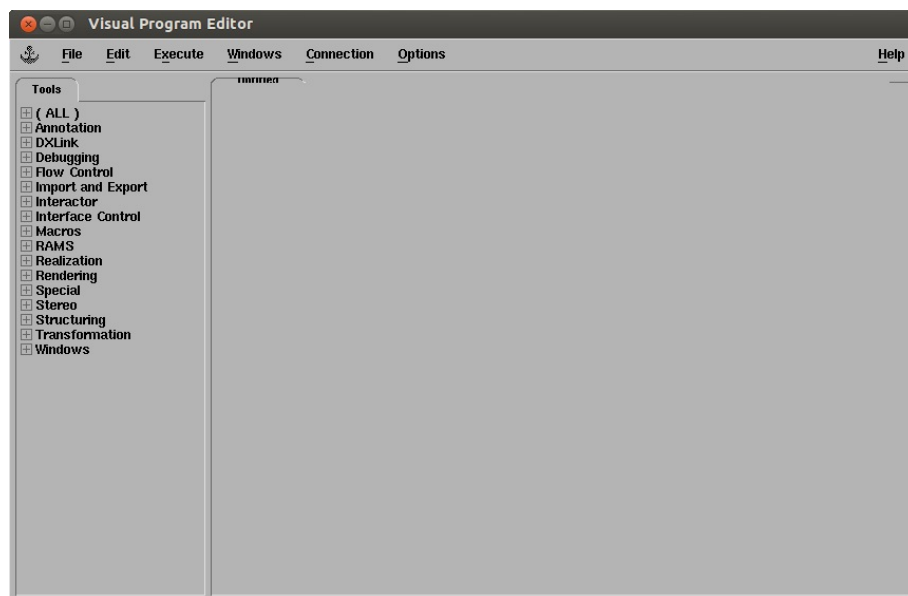


Figura 3: Ventana de trabajo del Visual Program Editor.

Para procesar y visualizar datos con OpenDX, debemos abrir la ventana del “Visual Program Editor” desde el menú principal. Se puede elegir entre “New Visual Program” ó “Edit Visual Programs”. Para ilustrar el uso básico del editor, abramos un programa nuevo. La interfaz se divide en dos regiones: a la izquierda tenemos un menú desplegable donde seleccionaremos los bloques a usar en el programa, y a la derecha se encuentra el área principal

de trabajo.

Ilustraremos con un ejemplo simple la creación de un programa para la representación gráfica de los datos proporcionados por el programa `funcion-2d.c`. Todo programa tipo OpenDX debe comenzar con un módulo **Import**, que se encarga de leer los datos de un fichero de entrada, y debe terminar con un módulo **Image**, que imprime por pantalla los datos procesados por el software OpenDX. Para colocar los bloques en la ventana de trabajo, podemos desplegar la pestaña “ALL” en el menú de la izquierda, y seleccionar a continuación con el ratón el bloque a incluir. Una vez hecho esto, colocando el ratón en la ventana de trabajo, y haciendo click, se coloca el bloque deseado.

Para nuestro ejemplo, colocaremos, como se muestra en la figura siguiente, los bloques **Import** (lectura de datos), **AutoColor** (que colorea los datos automáticamente según su valor) e **Image** (salida por pantalla). En caso de equivocarnos, podemos seleccionar cualquier bloque, y con la opción “Delete” del menú “Edit”, borrarlo. Una vez colocados los bloques, debemos realizar una segunda tarea fundamental, que es conectar los bloques entre sí. En el lenguaje de programación DX, los bloques procesan datos, que se transfieren de un bloque a otro a través de “tuberías”. Para conectar bloques, simplemente debemos hacer click en una de las pestañas de un bloque, y arrastrar con el ratón hasta la pestaña de otro bloque.

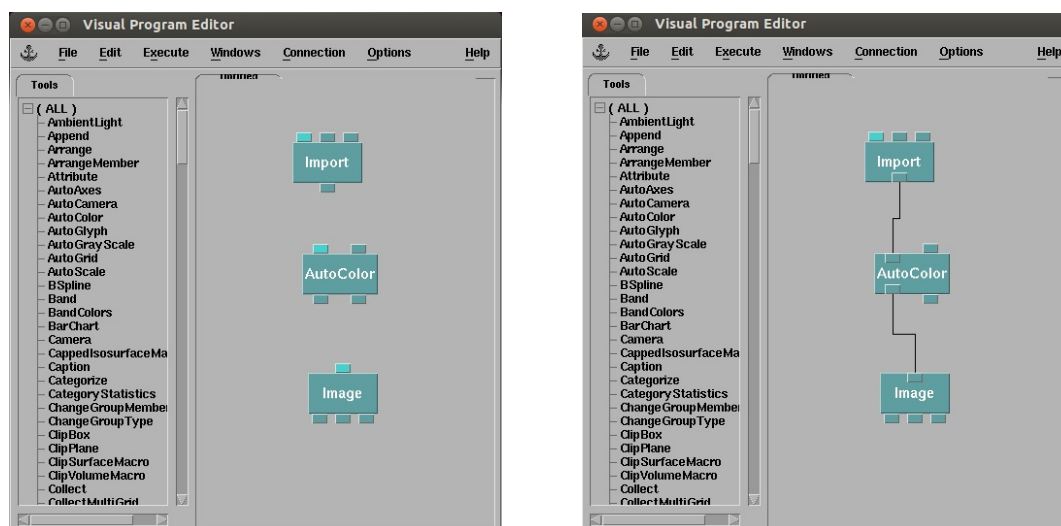


Figura 4: Creación de un programa sencillo en OpenDX.

En nuestro ejemplo, tal como se muestra en la figura, conectaremos el bloque **Import** al bloque **AutoColor** y el bloque **AutoColor** al bloque **Image**. Un vez hecho esto, lo único que nos queda hacer para que el programa sea funcional es **configurar las opciones del bloque Import**. Para ello, hacemos doble click en el mismo, con lo cual se abre una nueva ventana de opciones. En la misma, en el apartado “name”, debemos escribir el nombre del archivo de tipo `.general` (datos.general) que generamos anteriormente a partir de nuestro archivo de datos `datos.dat`. Es esencial, para evitar problemas con la ruta de directorios, que hayamos abierto el programa OpenDX en el mismo directorio donde se encuentran los datos. En caso contrario, debemos añadir la ruta completa al archivo `.general` con los datos.

Hecho todo lo anterior correctamente, sólo queda ejecutar nuestro programa DX para visualizar el resultado. Para ello, debemos hacer click en la opción “Execute on change” dentro de la pestaña “Execute”. Se abrirá una nueva ventana (**Image**) donde podemos ver que OpenDX representa los valores de la función bidimensional de acuerdo con un código de colores.

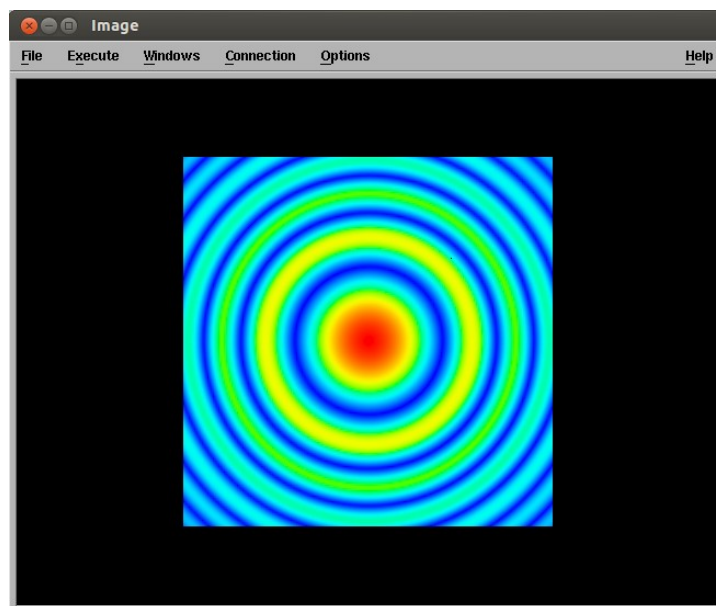


Figura 5: Ventana de visualización de resultados.

Desde la ventana de visualización, podemos salvar la imagen obtenida en diversos formatos gráficos (tif, postscript, etc...). Para ello, en la pestaña “File” debemos seleccionar la opción “Save Image”, y a continuación elegir el nombre del archivo con la imagen, así como su ubicación. No debemos olvidar hacer click en la pestaña “Save current” para que el programa guarde la imagen obtenida. En el caso de imágenes tridimensionales (que obtendremos más adelante), dentro del menú “Options” debemos abrir la pestaña “Mode” para controlar con el ratón el tamaño (opción “zoom”) y la orientación (opción “rotate”) de la imagen.

Mejoraremos ahora la apariencia de la imagen mediante el uso de algunos bloques de programación nuevos.

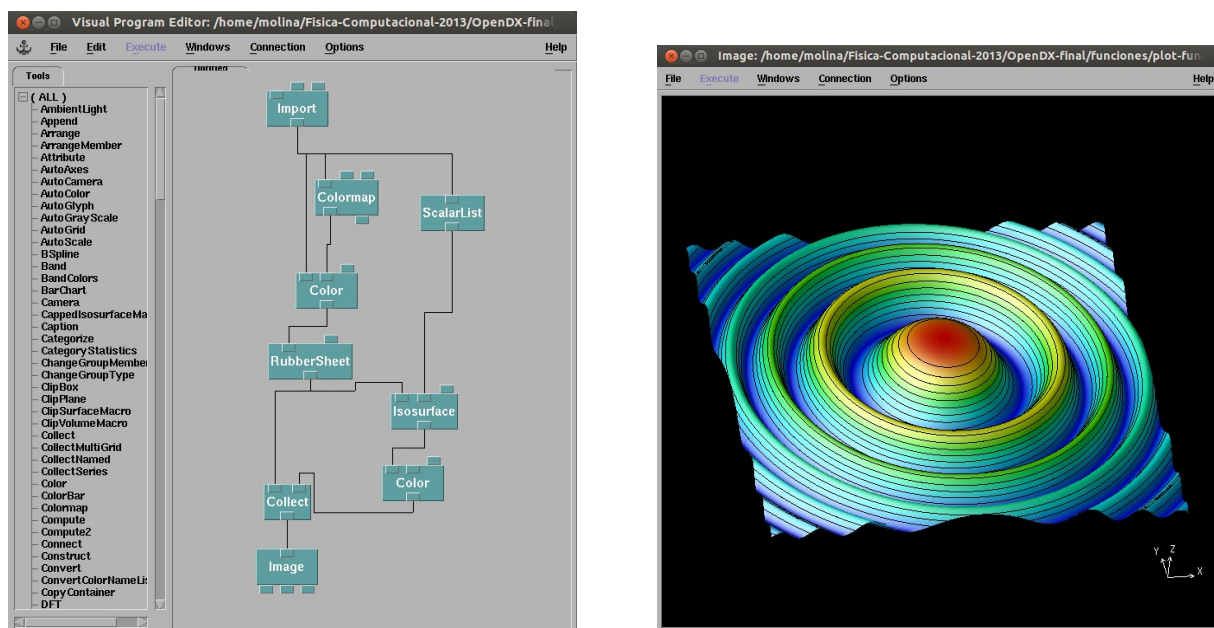


Figura 6: Programa “plot-funcion-2d.net” y su correspondiente resultado

En la página moodle de la asignatura, puede descargarse el programa de ejemplo “plot-funcion-2d.net”, con el diagrama de bloques mostrado en la figura 6. Para ejecutar este programa, se debe tener cuidado en reeditar la información sobre la ubicación del fichero de entrada de datos .general. Una vez ejecutado, el resultado es una representación tridimensional de los datos introducidos. El efecto 3D se obtiene incluyendo el bloque “RubberSheet”. Además del efecto 3D, se ha incluido en nuestro gráfico un cierto número de “curvas de nivel” (las líneas negras) que representan los puntos donde nuestros datos tienen un cierto valor constante. El conjunto de valores se define en el bloque “ScalarList”, al cual sigue el bloque “Isosurface”, que dibuja las líneas negras. El ejemplo mostrado ilustra también la forma de utilizar el bloque “Collect” para unir diversos elementos (superficie con efecto 3D y curvas de nivel) en el mismo gráfico. Finalmente, debemos avisar de que, en caso de que al ejecutar el programa sólo se vea una pantalla negra, se debe clicar la opción “Reset” dentro de la pantalla de la imagen. A veces ocurre, al ejecutar un programa, que la perspectiva está en un ángulo incorrecto y eso hace que no se vea la imagen.

Sección 3

Datos tridimensionales. Isosuperficies

Veremos ahora un par de ejemplos que ilustran la forma de representar datos tridimensionales en OpenDX. Nos referimos en este caso a datos que dependen de las coordenadas espaciales x , y , z , los cuales son representados por valores numéricos en una grid tridimensional (imaginemos, por ejemplo, que queremos visualizar el potencial eléctrico creado por una distribución de cargas en 3 dimensiones).

En primer lugar, en la página moodle, descarguemos y compilemos el programa `funcion-3d.c`. Este programa genera una matriz tridimensional de datos con los valores equiespaciados de la función

$$f(x, y, z) = \cos^2(x) * \cos^2(y) * \cos^2(z) * e^{-(x^2+y^2+z^2)/(4\pi^2)}$$

Como ya comentamos anteriormente, a la hora de ordenar los datos, el triple bucle anidado del programa que calcula y escribe los datos debe de estar estructurado de forma que la coordenada x corresponda al bucle más externo y la coordenada z al bucle más interno. Con tal precaución, para generar ahora el fichero de formato .general simplemente se debe añadir el número de puntos según la dirección z , así como el intervalo de datos en tal dirección.

Tras haber generado el fichero de datos `datos-funcion-3d.dat` y el fichero de formato `datos-funcion-3d.general`, abramos a continuación el programa `plot-funcion-3d-1.net` (descaragable también en la página moodle). Tras editar la ruta al archivo .general en el módulo Import, podemos ejecutar este programa y visualizar el resultado. Rotando la figura, deberíamos poder visualizar algo similar a lo que se muestra en la figura 7.

El programa, en este caso, utiliza el módulo “Isosurface” para representar tridimensionalmente las superficies con un valor constante de la función $f(x, y, z)$. Para una mejor representación de los valores de la función, hemos elegido en este caso representar las isosuperficies para tres valores diferentes de la función (0.1, 0.4 y 0.6). En cada módulo Isosurface, se debe especificar dentro del apartado “Value” el valor de la isosuperficie a representar. Por otro lado, en cada módulo “Color”, se debe especificar el color con el que dibujaremos cada isosuperficie. Finalmente, para conseguir el efecto que deseamos, debemos también (dentro del módulo Color,

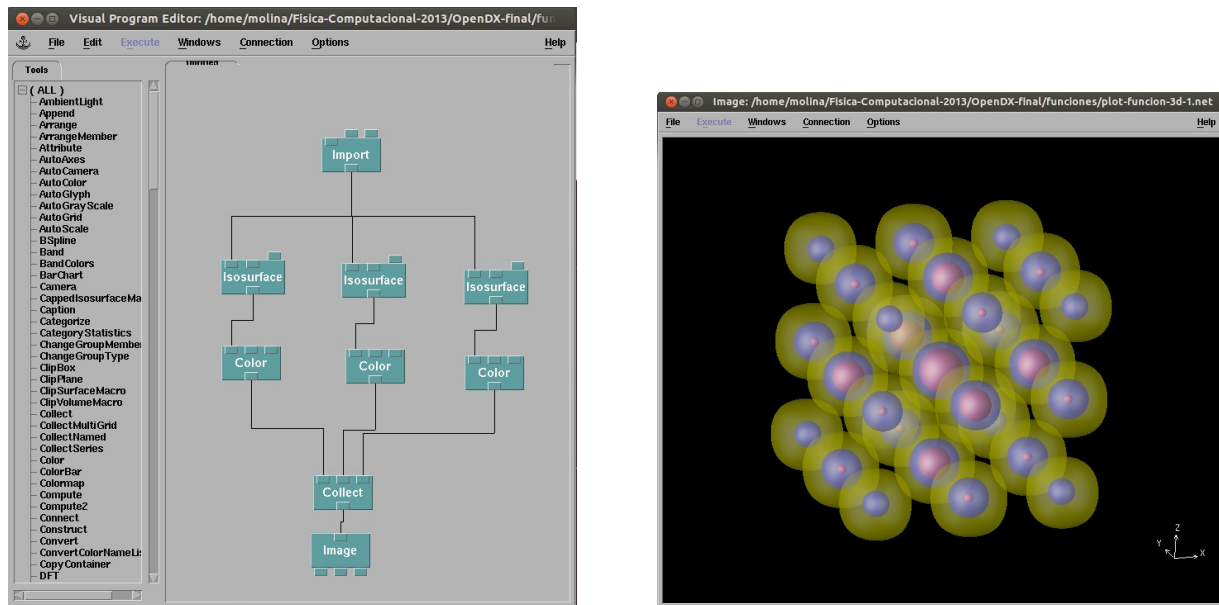


Figura 7: Programa “plot-funcion-3d-1.net” y su correspondiente resultado

que colorea las isosuperficies) especificar valores variables de opacidad (parámetro Opacity) para cada una de las tres isosuperficies. De esta manera, las isosuperficies se dibujan con un cierto grado de transparencia, que permite ver lo que se esconde dentro de cada una de ellas.

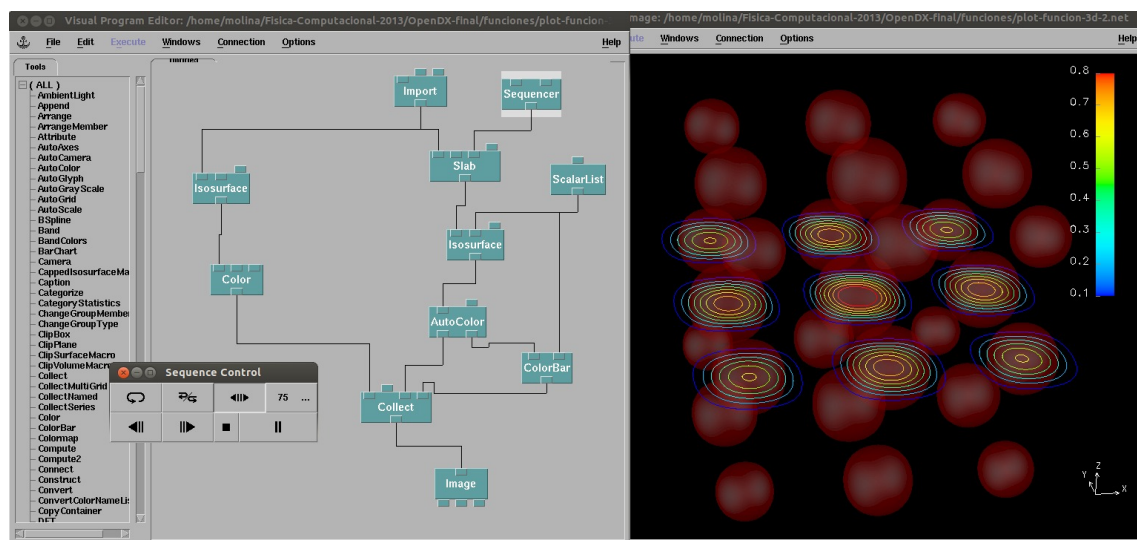


Figura 8: Programa “plot-funcion-3d-2.net” y su correspondiente resultado

El programa “plot-funcion-3d-2.net”, cuyo programa y resultado final se muestran en la figura 8, muestra una forma alternativa muy útil de visualización de datos escalares tridimensionales. Mediante el módulo Slab, cortamos los datos tridimensionales proyectándolos en un plano con una valor de la coordenada z constante. Seguidamente, utilizamos los módulos Isosurface, ScalarList y AutoColor para dibujar curvas de valores de $f(x, y, z)$ constante en el plano elegido. La curvas de nivel son representadas con un código de colores, cuya leyenda se incluye mediante el módulo ColorBar.

Además, en este programa mostramos un ejemplo interesante del empleo del módulo Se-

quencer. Haciendo doble click en ese módulo, se abre una ventana de control, que nos permite modificar interactivamente los valores numéricos de salida de ese bloque. Dado que hemos conectado el módulo Sequencer a la tercera pestaña del módulo Slab, que controla la altura del plano donde proyectamos los datos, avanzando adelante y atrás con el control de secuencia se puede cambiar instantáneamente de plano.

Sección 4

Otros ejemplos interesantes

En la página moodle de la asignatura, se han colgado también otros ejemplos de uso del software OpenDX. En la figura 9 puede verse el diagrama del programa “onda-movie.net” y el resultado de su ejecución.

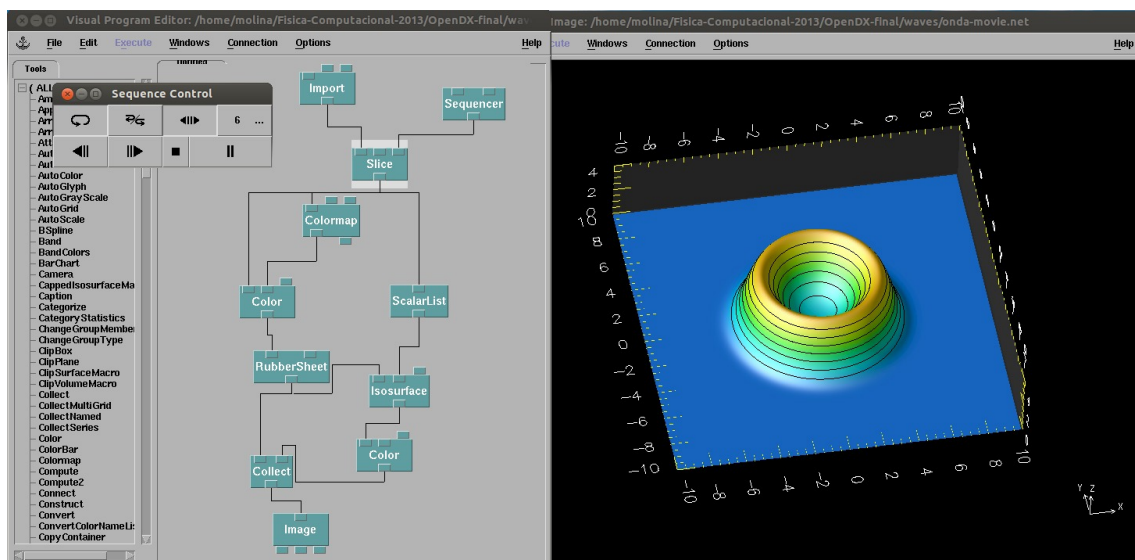


Figura 9: Programa “onda-movie.net” y su correspondiente resultado.

En este programa, procesamos varios conjuntos de datos bidimensionales, donde cada uno de ellos representa la forma en que una onda bidimensional se propaga a lo largo del tiempo. A la hora de introducir estos datos, los mismos se han colocado (archivo datos-onda-2d.dat) en una matriz tridimensional de datos, donde la primera coordenada representa el tiempo, y las segunda y tercera coordenadas representan las coordenadas (x,y) donde calculamos los valores de la onda bidimensional. En este ejemplo, se leen 25 “fotogramas” de la evolución de la onda.

A estructura del programa onda-movie.net es similar a la del ejemplo ya visto para la representación de datos bidimensionales, con un añadido. Se ha empleado el módulo “Slice” para cortar una sección bidimensional de nuestro conjunto de datos (según la dimensión número 0, que representa el tiempo). La tercera pestaña del bloque Slice, que representa la sección que se escoge, es controlada por el módulo Sequencer, cuyos valores varían desde 0 a 24. El resultado de la ejecución, si vamos avanzando fotograma a fotograma la película de datos con el módulo Sequencer, es una visualización interactiva de la evolución temporal de la onda.

Finalmente, debemos mencionar que es posible, mediante la opción “Continuos Saving” de la ventana “Save Image”, generar automáticamente una sucesión de todas las imágenes de la película y guardarlas en un fichero con formato MIFF. Posteriormente, mediante el conversor de formatos `convert` de Linux, es posible reconvertir el archivo MIFF en un archivo de tipo GIF animado.

El último ejemplo que comentaremos es considerablemente más complejo; para su ejecución, deben descargarse de la página moodle los archivos `benceno-orbitales.dat.gz` (el cual debe descomprimirse tras su descarga), `benceno-orbitales.general`, `benceno-orbitales.net` y `geometria.dx`. Tras editar el campo “Name” del módulo Import del programa `benceno-orbitales.net`, al ejecutar el mismo se obtiene el resultado mostrado en la figura 10.

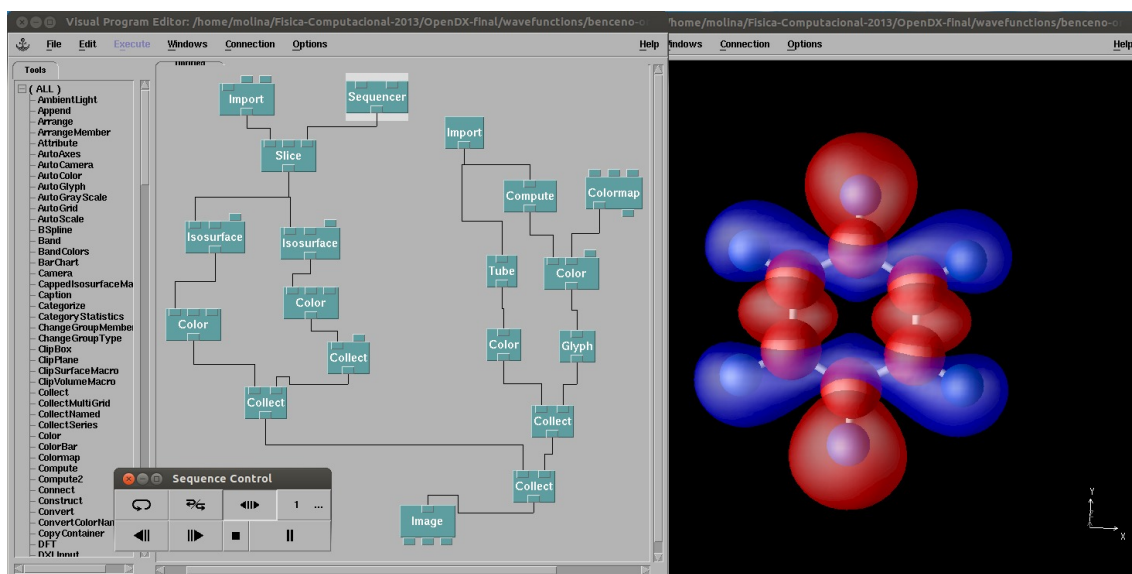


Figura 10: Programa “onda-movie.net” y su correspondiente resultado.

Este programa lee un fichero cuádr dimensional de datos (`benceno-orbitales.dat`) donde se ha almacenado una sucesión de datos escalares tridimensionales correspondientes a varios orbitales (ó funciones de onda) de la molécula de benceno (C_6H_6). Los diferentes orbitales pueden visualizarse interactivamente mediante el control del módulo Sequencer. Para cada orbital, el programa representa dos isosuperficies tridimensionales de valor constante, una positiva y otra negativa, las cuales dan idea de la distribución electrónica y del enlace químico en cada orbital de la molécula. Además, el programa lee los datos estructurales de la molécula (almacenados en un formato especial en el archivo `geometria.dx`) y dibuja una representación tridimensional de la misma, con sus 6 átomos de carbono, sus 6 átomos de hidrógeno, y los correspondientes enlaces entre los mismos.