Proyecto de Modelado Matemático I:
Visualización 3D en VisIt

Paula Camila Wandurraga Sanabria

Maestría en Matemática Aplicada
Grupo de Investigación en Relatividad y Gravitación
Universidad Industrial de Santander

Abstract

En el presente trabajo se realiza

1 Introducción

Aquí va la intro

2 Objetivos

2.1 Objetivo General

Visualizar los datos 3D resultantes de simulaciones de propagación de ondas en la atmósfera solar, con VisIt.

2.2 Objetivos Específicos

- Conocer los formatos de datos que recibe VisIt.
- Crear un convertidor de formato que pase los datos de las simulaciones a un formato válido para VisIt.
- Escribir un script en bash para hacer más eficiente el proceso.
- Visualizar los datos por bloques en VisIt.

3 Planteamiento del Problema

Anteriormente, los datos eran extraídos por bloques y visualizados con un código de Python para MayaVi. El proceso de visualización en MayaVi se complica, ya que la aplicación se buggea fácilmente y se pierde el trabajo hecho en la sesión. Por esta razón, se decidió cambiar el visualizador de datos 3D. En medio de la búsqueda, se quiso aprender a visualizar en VisIt, ya que es ampliamente utilizada en el ámbito científico, más específicamente en física solar. VisIt es una herramienta de código libre, está diseñada para soportar datos de gran tamaño, permite visualizar campos escalares y vectoriales y realizar operaciones entre conjuntos de datos.

VisIt recibe una gran cantidad de formatos de texto, como archivos AUX, HDF4, HDF5, AMR, M3D, OVERFLOW, algunos tipos de archivo ASCII, entre otros. Los datos salvados en las simulaciones están en un formato ASCII no compatible con VisIt, por tanto, se podía cambiar la manera de salvar datos y volver a realizar las simulaciones o traducir los datos a alguno de los formatos válidos para VisIt. Debido a que las simulaciones que actualmente se están trabajando, son un grupo de 18 corridas que demoran aproximadamente 2 días cada una, no es viable volver a realizarlas, es mejor fabricar el convertidor de formatos.

Cabe aclarar que cada simulación arroja un único archivo de datos 3D que contiene todas las variables (malla, densidad, componentes de la velocidad, presión, componentes del campo magnético) y todos los pasos de tiempo que se simularon. Por esto, se utiliza un código escrito en Fortran para extraer diferentes bloques de tiempo del archivo y leerlos en el código de Python para

MayaVi.

4 Código

Inicialmente, la idea era crear un convertidor de ASCII a HDF5 para poder visualizar. Este código (ver figura 1) ya estaba hecho, se encontró en el siguiente enlace: Convertidor de ASCII a HDF5, y funciona. Aun así, no fue satisfactorio para el proyecto, ya que el sabor de HDF5 que arroja como resultado el código, no está entre los posibles formatos de datos para VisIt.

```
from __future__ import print_function
import os
from astropy.io import ascii

def ascii2hdf5(inputfile, outputfile, clobber=False, overwrite=True,
               verbose=False):
    """Convert a file to hdf5 using compression and path set to data"""
    if verbose:
        print('converting {} to {}'.format(inputfile, outputfile))

    tbl = ascii.read(inputfile)
    try:
        tbl.write(outputfile, format='hdf5', path='data', compression=True,
                  overwrite=overwrite)
    except:
        print('problem with {}'.format(inputfile))
        return

    if clobber:
        os.remove(inputfile)
        if verbose:
            print('removed {}'.format(inputfile))

    return
```

Figure 1: Código que toma un *inputfile* en formato ASCII y lo convierte en un *outputfile* en formato HDF5.

Como solución alternativa, se hizo un código basado en un código de Python que genera archivos ASCII compatibles con el visualizador: Generador de archivos en formato permitido. El código creado funciona como extractor y convertidor al mismo tiempo, tomando los datos en bruto y generando datos organizados que VisIt pueda leer.

```
import sys
import numpy as np

# Se importa el archivo con los datos correspondientes
# a las dimensiones y tiempos de las corridas
filedata = open('1input_plots.par', 'r')

lines = filedata.readlines()
# Se asignan la variables del archivo filedata en
# las variables correspondientes
xmin = float(lines[1].split('=')[1])
xmax = float(lines[2].split('=')[1])
ymin = float(lines[3].split('=')[1])
ymax = float(lines[4].split('=')[1])
zmin = float(lines[5].split('=')[1])
zmax = float(lines[6].split('=')[1])
Nxx = int(lines[7].split('=')[1])
Nyy = int(lines[8].split('=')[1])
Nzz = int(lines[9].split('=')[1])
Ntt = int(lines[10].split('=')[1])
courant = float(lines[11].split('=')[1])
every3D = float(lines[14].split('=')[1])

# Se cierra el archivo
filedata.close()

# Se calculan los pasos espaciales y temporales
dx  = (xmax - xmin)/float(Nxx)
dy  = (ymax - ymin)/float(Nyy)
dz  = (zmax - zmin)/float(Nzz)
dt = courant * min(dx,dy,dz)
```

Figure 2: Primeras líneas

5 Resultados

Los archivos generados por el código de Python creado, fueron

6 Conclusiones

Se cumplió safisfactoriamente con los objetivos trazados para el proyecto, ya que se lograron visualizar los datos del código MAGNUS con VisIt

7 Referencias