8 Gennaio 2023

Cari amici, il progetto SnagPy è andato avanti ed ora posso dire che la prima fase, quella di definire gli algoritmi di base, è pressocché finita, e quindi vorrei fare il punto della situazione.

I nuovi file, con anche la user guide aggiornata, la trovate al solito <https://github.com/SerFrasca/SnagPy>

In SnagPy ci sono tre livelli: quello “informatico” (strutture base), quello “matematico” (per esempio filtri e bsd) e quello “fisico” (gli elementi teorici e l’analisi dati effettiva). Questa prima fase è stata essenzialmente “informatica”. Lo sviluppo è stato fatto prevalentemente con i moduli GD, ML\_PY e BASIC.

GD definisce la classe principale di Snag. In GD i gd possono essere creati, modificati, manipolati, plottati ecc. In particolare ci sono alcune funzioni che semplificano il plotting di gd (o anche array).

ML\_PY contiene le funzionalità per leggere i file prodotti da Matlab, sia nel formato v7, che nel formato v7.3. In particolare possono leggersi correttamente anche i bsd, con la loro complessa struttura.

BASIC contiene il grosso delle function “informatiche”. Ci sono varie funzioni per gestire l’input e l’output su file in vari formati; in particolare c’è la gestione del formato HDF5 che, come suggeriva Marco, sarà il formato principale per i dati prodotti da SnagPy. Ci sono poi varie funzioni per semplificarsi la vita con le liste e i dictionary, per esempio per mostrare la struttura di dictionary complessi ed accedere ai loro dati. Sono quindi introdotte due comode strutture, i simple dict (dictionary semplificati) e le snag\_table (molto simili alle table di Matlab. Ci sono poi funzioni per gestire i NumPy structured arrays (una cosa molto pythonica, che serve per esempio per trattare i dati provenienti da mat file formato v7). Infine qualche funzione generale utile e due funzioni che analizzano i moduli per la documentazione (una per un’analisi sintetica e una più approfondita.

Vi riporto un esempio di uso (da SnagPy\_UG, la user guide).

## **Plotting**

The main procedure to plot gds and mono-dimensional arrays is based on the use of, normally, four functions:

1) newfig that can define the dimension of the window. If the

parameter siz is a two element list, it defines the relative

enhancement of the horizontal and vertical dimension of the

window, if it is just a number it applies to both h and v.

Example: newfig([1.33,1])

2) plot\_helper that defines various aspects of the graph, e.g.:

mode : normal plot, steps, scatter plot

scale : linear or logatithmic scale for each axis.

grid : grid lines

fmt : format (color and texture)

linewid: line width

It creates a dictionary called P\_H.

3) plot\_gd general plotting function; the imput are:

ingd the input gd or array

P\_H the output of plot\_helper; if absent, the default

4) post\_plot defines title and labels.

The plot can be modified by the functions xlog, ylog, xlin, ylin,

xlim, ylim and others.

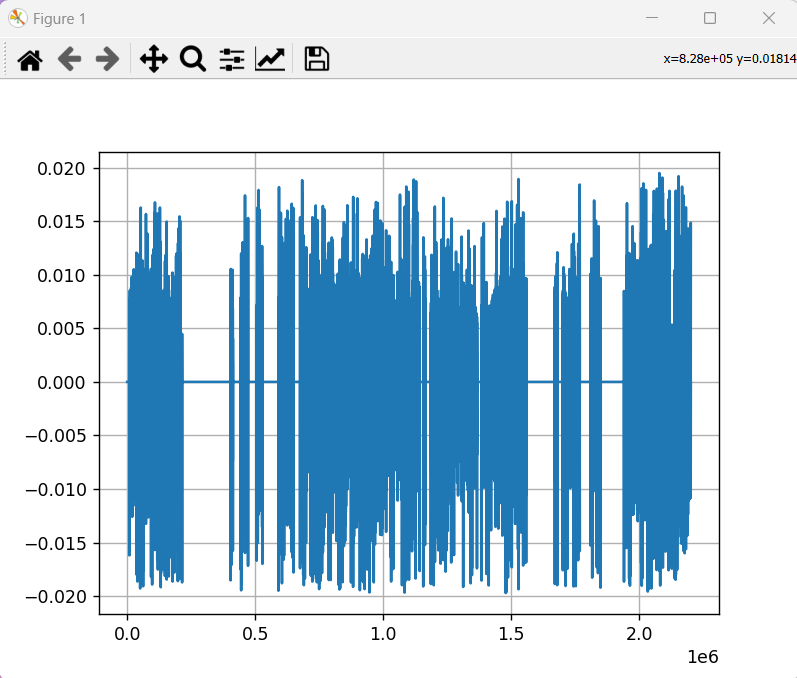
Example:

>>> l\_bsd=ML\_PY.gd\_lm7('L\_C02\_20170104\_0060\_0070\_O2\_tfstr.mat')

>>> GD.newfig()

>>> GD.plot\_gd(l\_bsd

and we have:



Then

>>> s=STAT.gd\_pows(l\_bsd,npiece=10,res=2)

>>> GD.newfig([1.33,1])

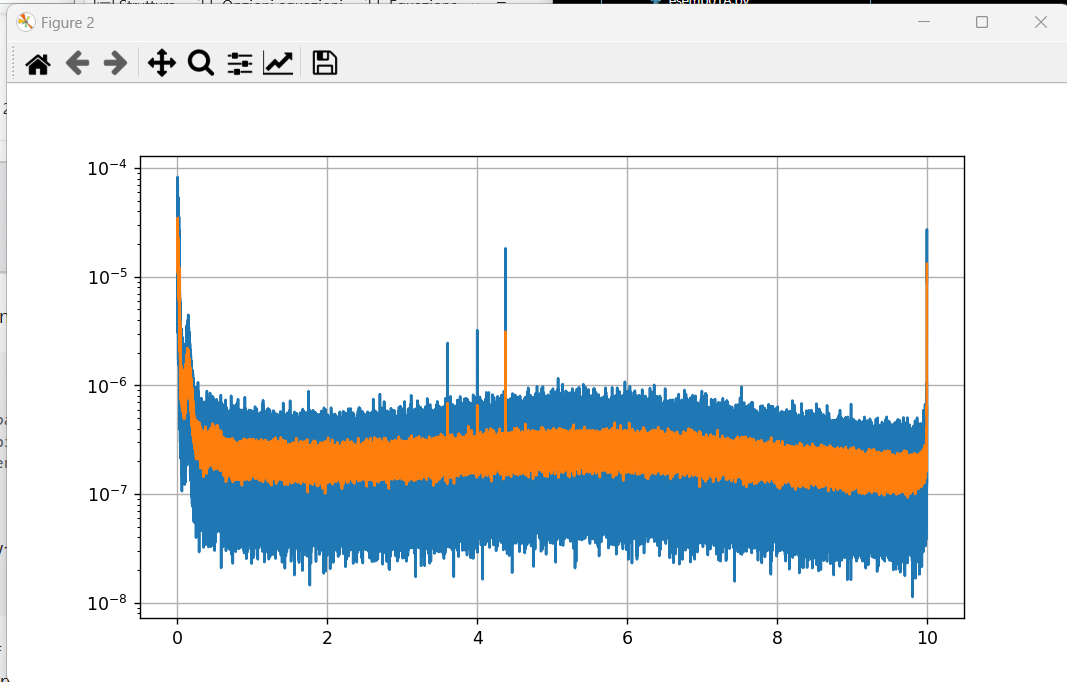
>>> P\_H=GD.plot\_helper(scale='lilo')

>>> GD.plot\_gd(s,P\_H)

>>> s=STAT.gd\_pows(l\_bsd,npiece=100,res=2)

>>> GD.plot\_gd(s,P\_H)

obtaining:



and zooming:

