Snag v2

Signal and noise for gravitational antannas

User Guide

http://grwavsf.roma1.infn.it/snag/

Table of contents

Introduction

Snag is a Matlab (C) object oriented toolbox for data simulation and analysis of signal and noises for gravitational antennas.
Basic requirements is the base Matlab 6 installation (no other toolbox required).
Matlab is available on a variety of operative systems (mainly Windows and Linux).

This is the User Guide. For all programming issues, please refer to the programming guide Snag2_PG.pdf .

Installation

The distribution is in a zip file.
a) Unzip the file where you want put the snag directory. This will create a snag directory and a few subdirectories (some of which, beginning by @, are class directories), and installs some m-files and a few mat and doc files.
b) Add to Matlab the path of Snag and of the subdirectories not beginning by @, excluding the "local" subdirectory. In some cases (for example Unix centralized Matlab installations) this can be difficult, so a snagpath.m file is provided (in the local directory): relocate the file (see point c)) and modify its first command line with the actual path; in this case you must run snagpath at the beginning of a snag session.
c) Copy the m-files contained in the "local" directory in a different (on path) location, for example in the snag directory. [This avoids the overwriting of the local files in case of upgrades.] Then modify the content of this file according to the path of the data files and other local choices.
d) Check the installation typing at the Matlab prompt

```
>> snag
```

or

```
>> data_browser
```

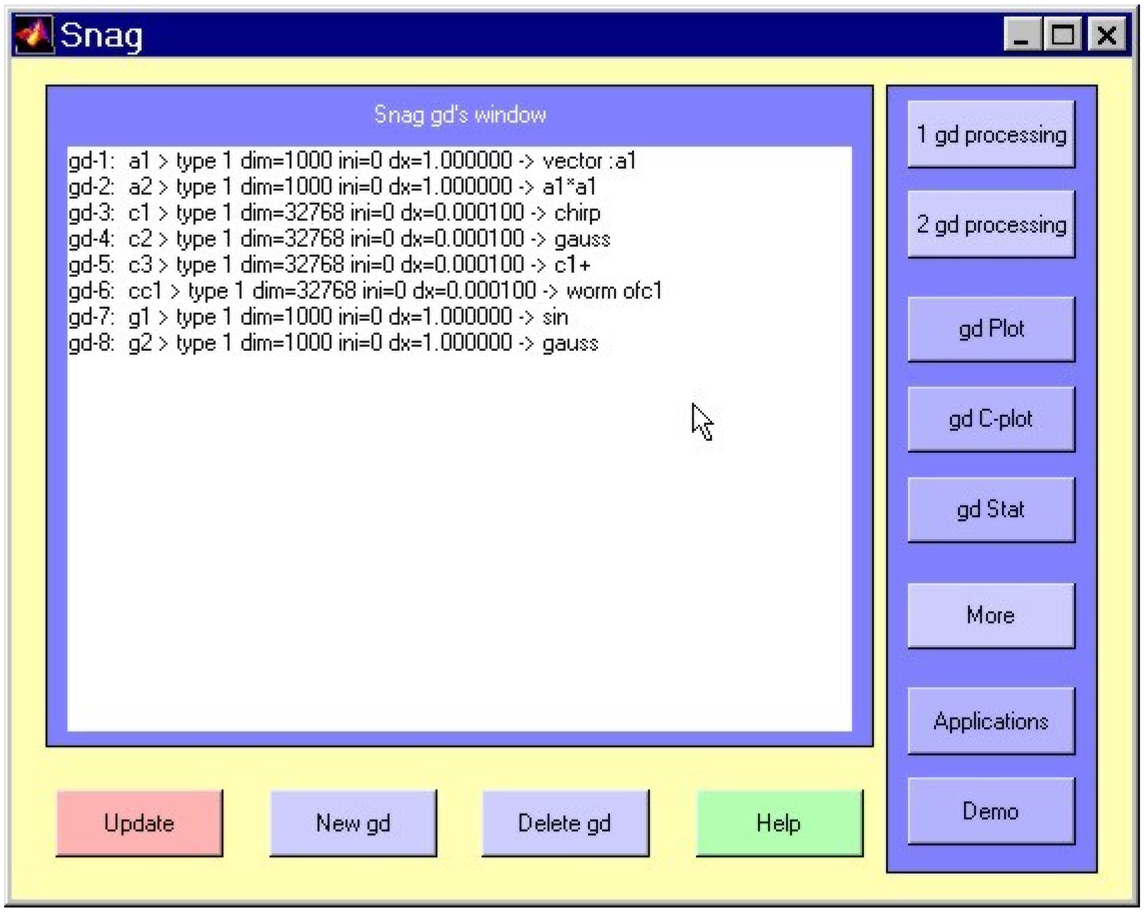
(that can be simplified by "db" if the alias db.m is active).

Part I - Snag GUIs

Here only some of the features of the GUI Snag applications are presented. Up to now only **snag** and **data_browser** are completely developed.

snag

snag provides a GUI access to the Snag functionalities. It can be used "stand-alone", or in conjunction with the normal Matlab prompt use of Snag . At the Matlab command prompt, type **snag .** A window appears:

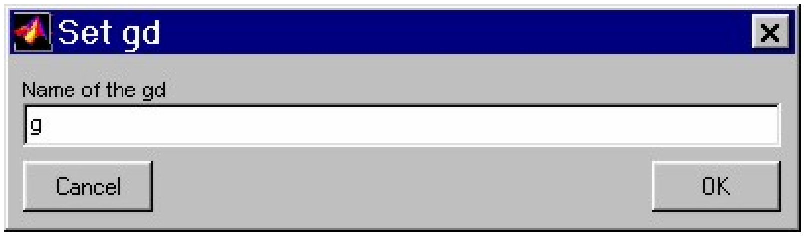


```
gd-1:  a1 > type 1 dim=1000 ini=0 dx=1.000000 -> vector :a1
gd-2:  a2 > type 1 dim=1000 ini=0 dx=1.000000 -> a1*a1
gd-3:  c1 > type 1 dim=32768 ini=0 dx=0.000100 -> chirp
gd-4:  c2 > type 1 dim=32768 ini=0 dx=0.000100 -> gauss
gd-5:  c3 > type 1 dim=32768 ini=0 dx=0.000100 -> c1+
gd-6:  cc1 > type 1 dim=32768 ini=0 dx=0.000100 -> worm ofc1
gd-7:  g1 > type 1 dim=1000 ini=0 dx=1.000000 -> sin
gd-8:  g2 > type 1 dim=1000 ini=0 dx=1.000000 -> gauss
```

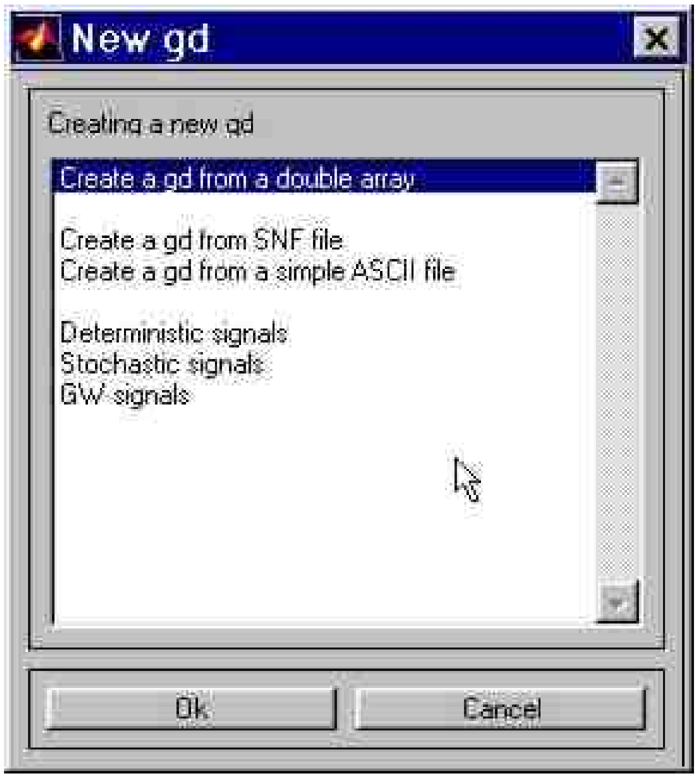
It has a text window, where are listed the **gd**s that have been created and some buttons.

The first one, the **Update** button, is used to update the content of the text window: just push it.

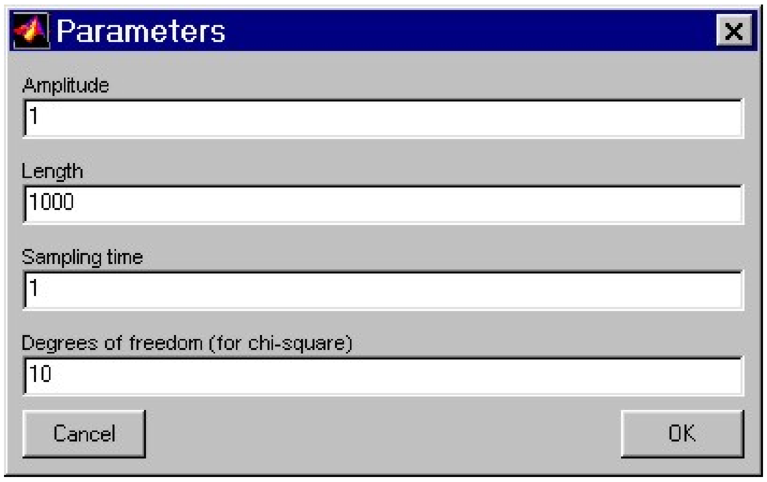
The **New gd** button is used to create a new **gd** from a Matlab double array, from files or from scratch; it opens some input dialog windows, the first of which asks for the name of the new gd



the second


New gd
Creating a new gd
Create a gd from a double array
Create a gd from SNF file
Create a gd from a simple ASCII file
Deterministic signals
Stochastic signals
GW signals
Ok
Cancel

asks for the type of data to put inside; it can be created "from a double array" (that must exist), "from a file" (in different formats) or computing different types of signals. For example, if the "stochastic signals" is chosen, a list dialog box appears, and if "White noise" is chosen,


Parameters

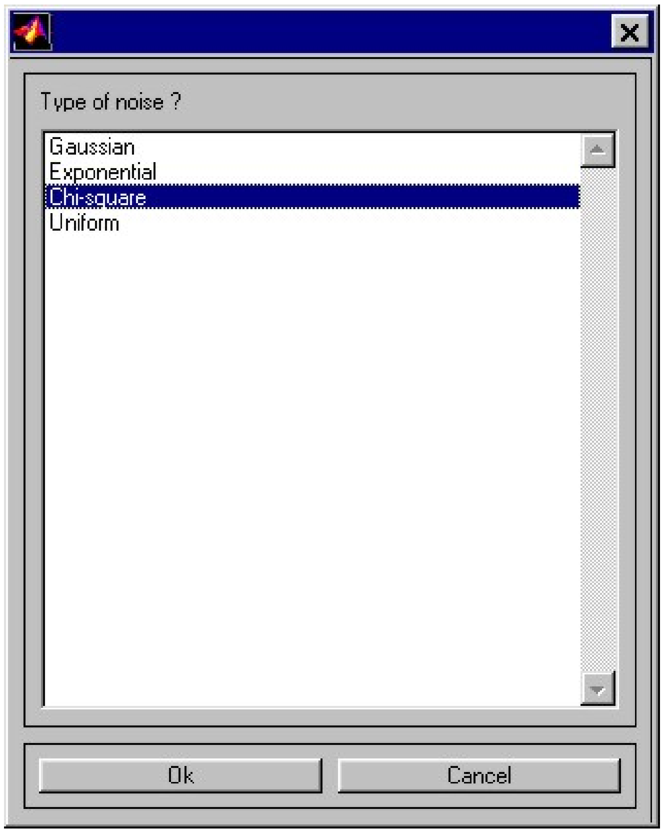
Amplitude
1

Length
1000

Sampling time
1

Degrees of freedom (for chi-square)
10

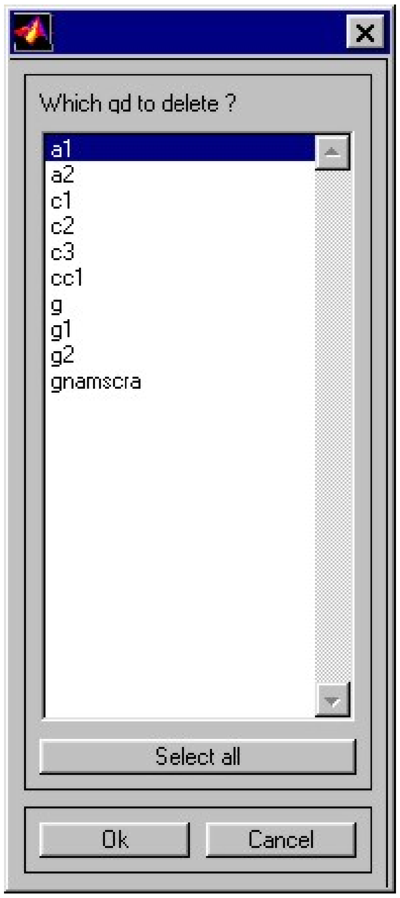
asks for some parameters, and then another window appears,


Type of noise ?

Gaussian
Exponential
Chi-square
Uniform

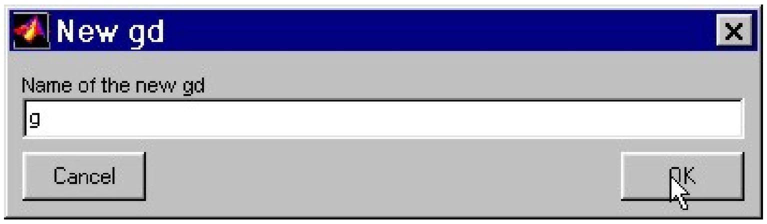
that asks for the type of noise to put inside. Similar behaviour for other menu choises.

The **Delete gd** button simply asks which **gd** do you want delete, with the window

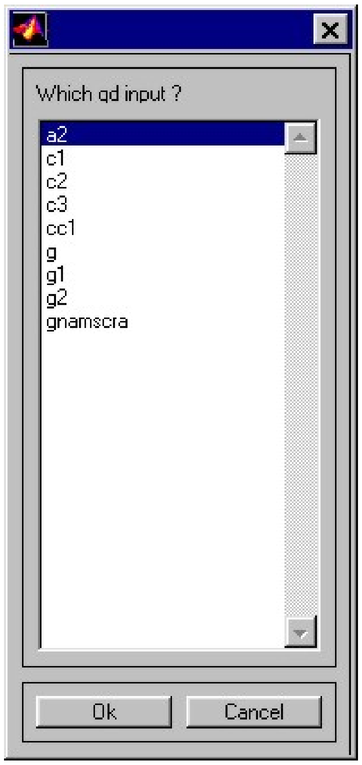


that lists the **gd**s.

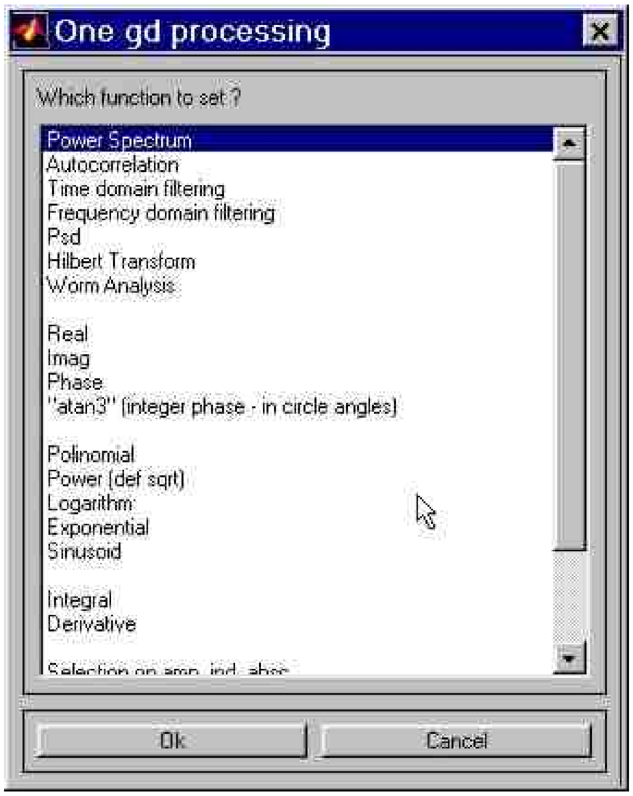
On the vertical bar there are some other buttons. The **1 gd processing** button allows the processing on a single **gd**, producing a new **gd**; it opens successively three windows,



asks for the name of the **gd** that will be produced,

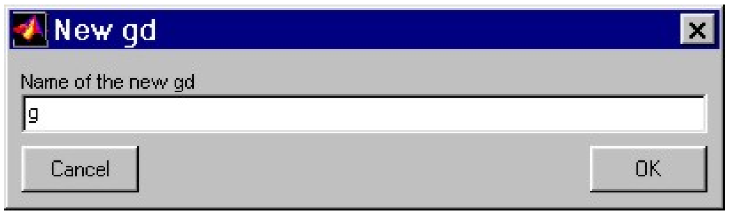


asks for the **gd** to be processed, and

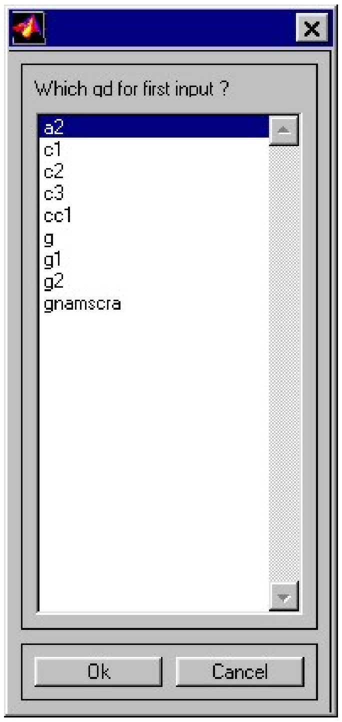


asks for the type of processing to be performed.

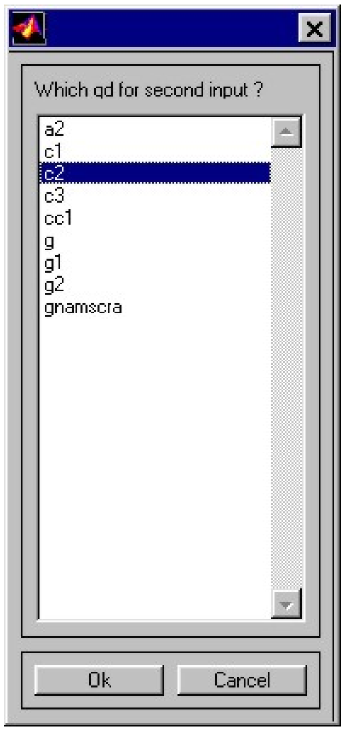
The **2 gd processing** button allows the processing on two **gd**s, producing a new **gd**; it opens successively four windows,



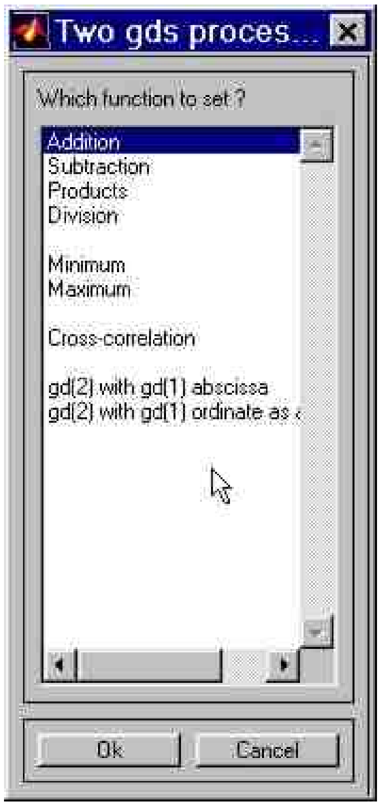
asks for the name of the output **gd,**



asks for the first input **gd**,

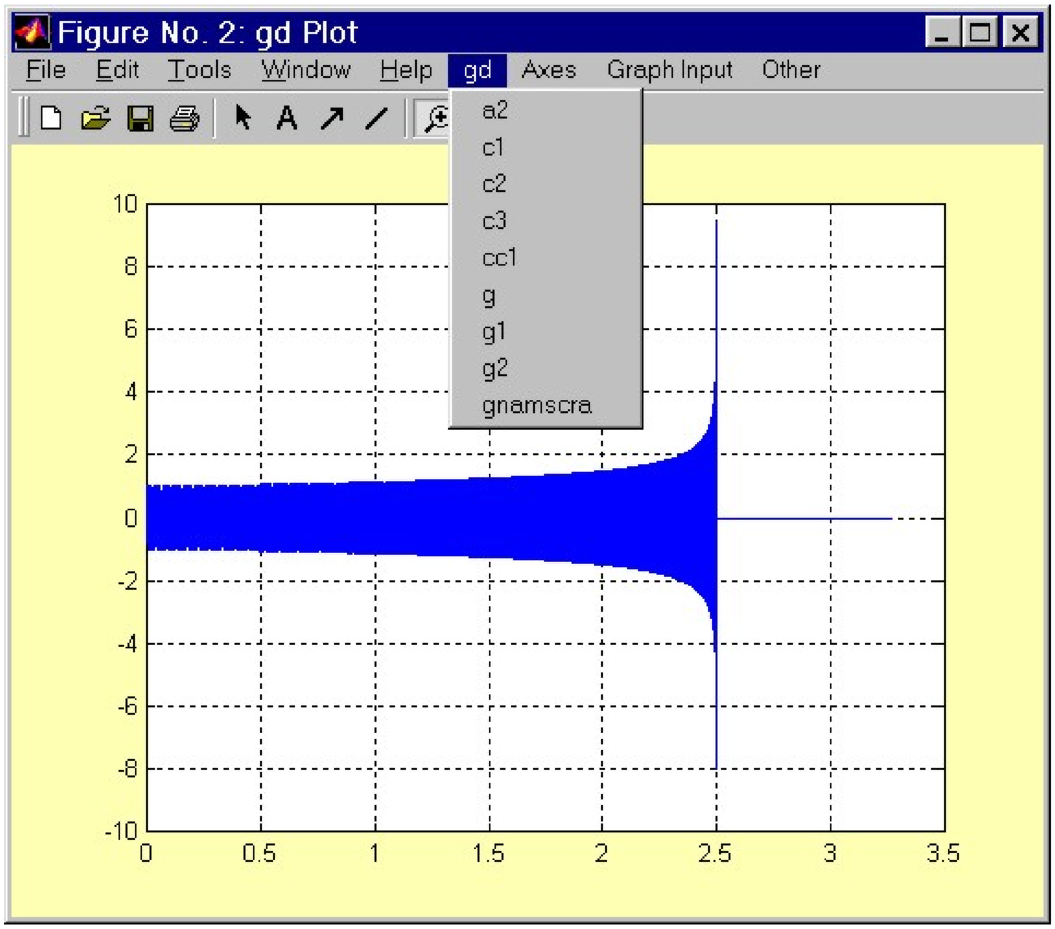


asks for the second input **gd,**

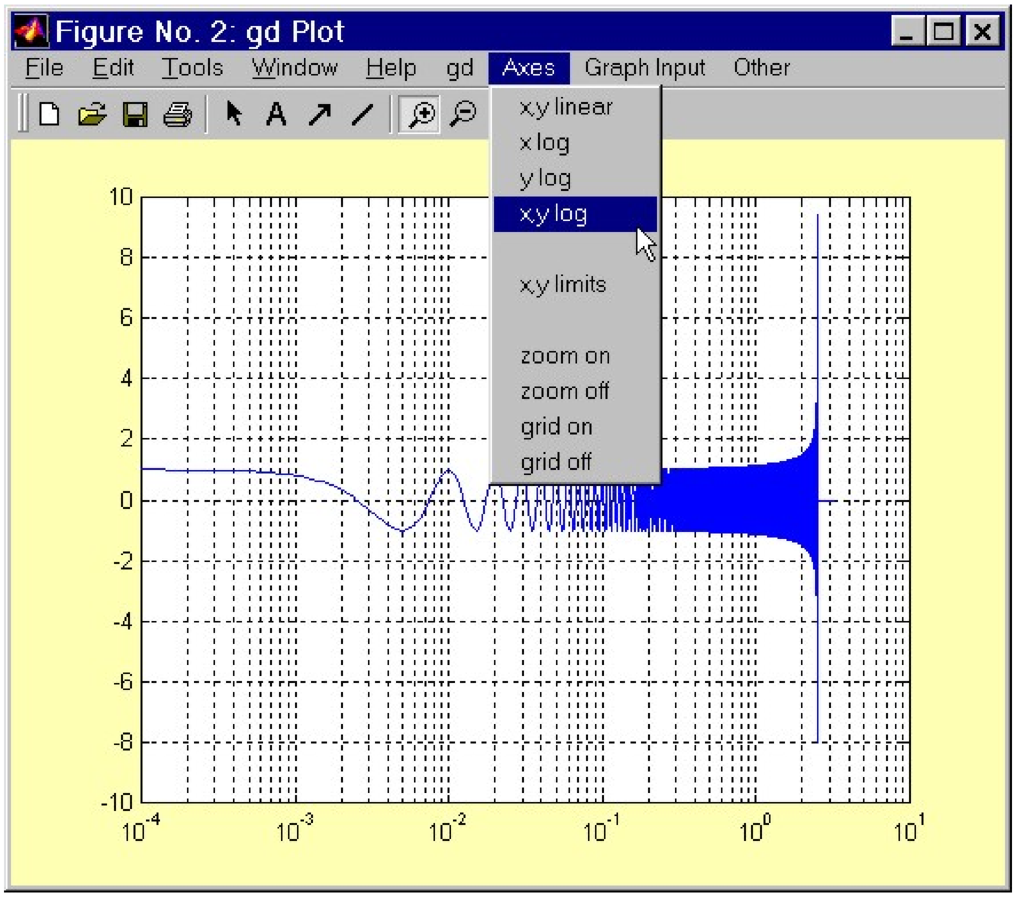


asks for the type of processing.

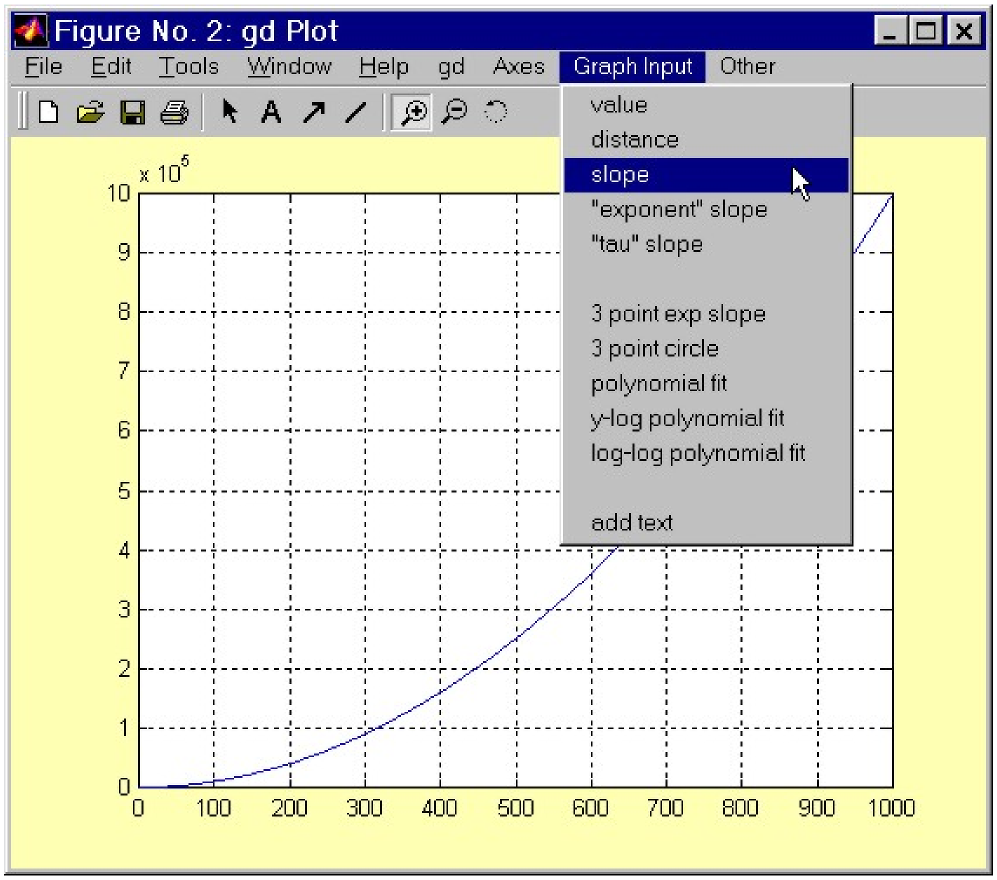
The **gd Plot** button opens a plotting facility



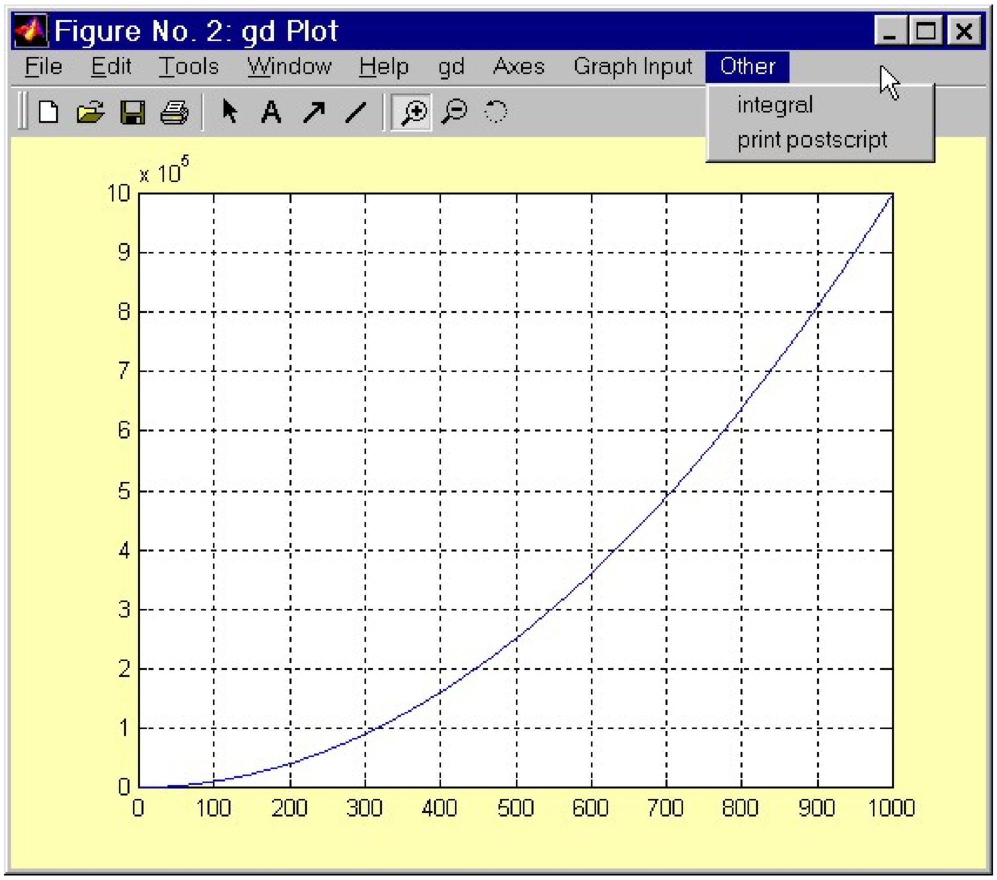
that uses the Matlab plotting function, with some added menus. One must chose, at the beginning the **gd** to be plotted (you can plot more **gd**s in the same graph); the **Axes** menu



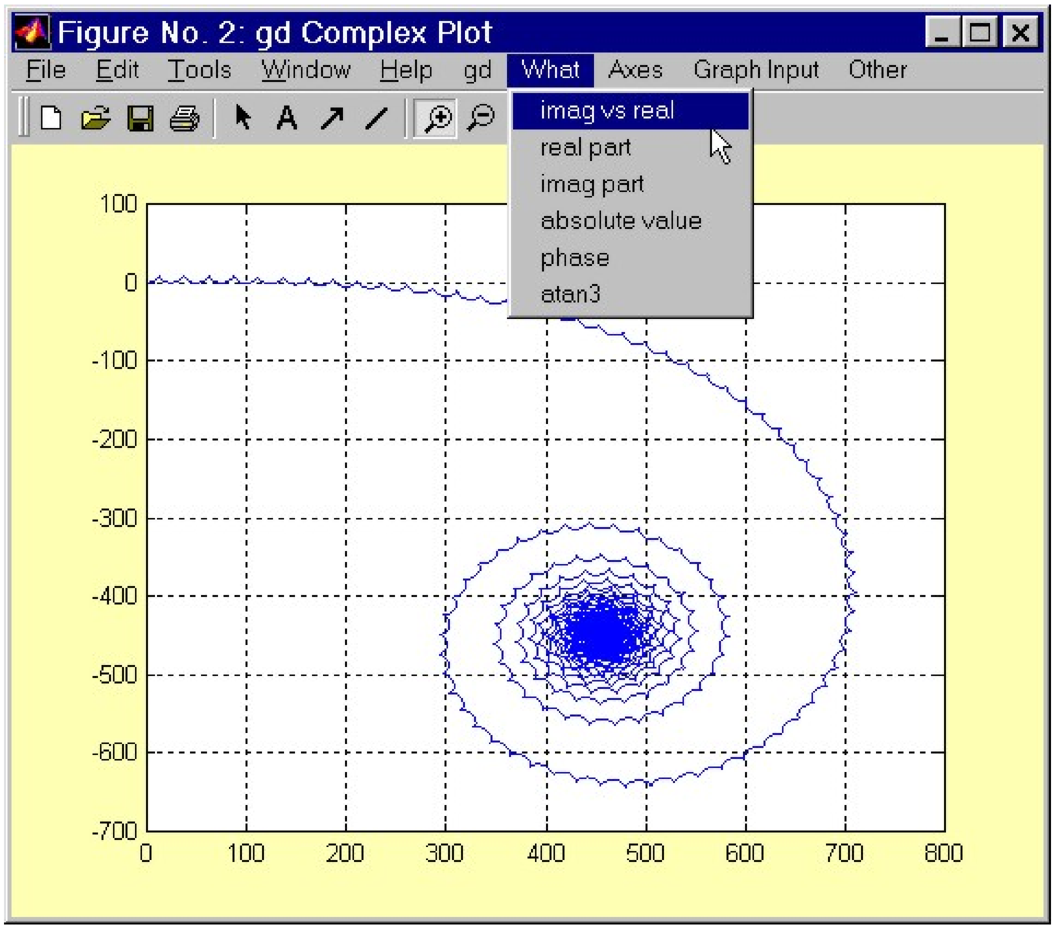
allows some choices on the axes; any choice of the axes resets the graph, so it can be used to do new graph (otherwise more plots goes in the same graph).
The **Graph Input** menu



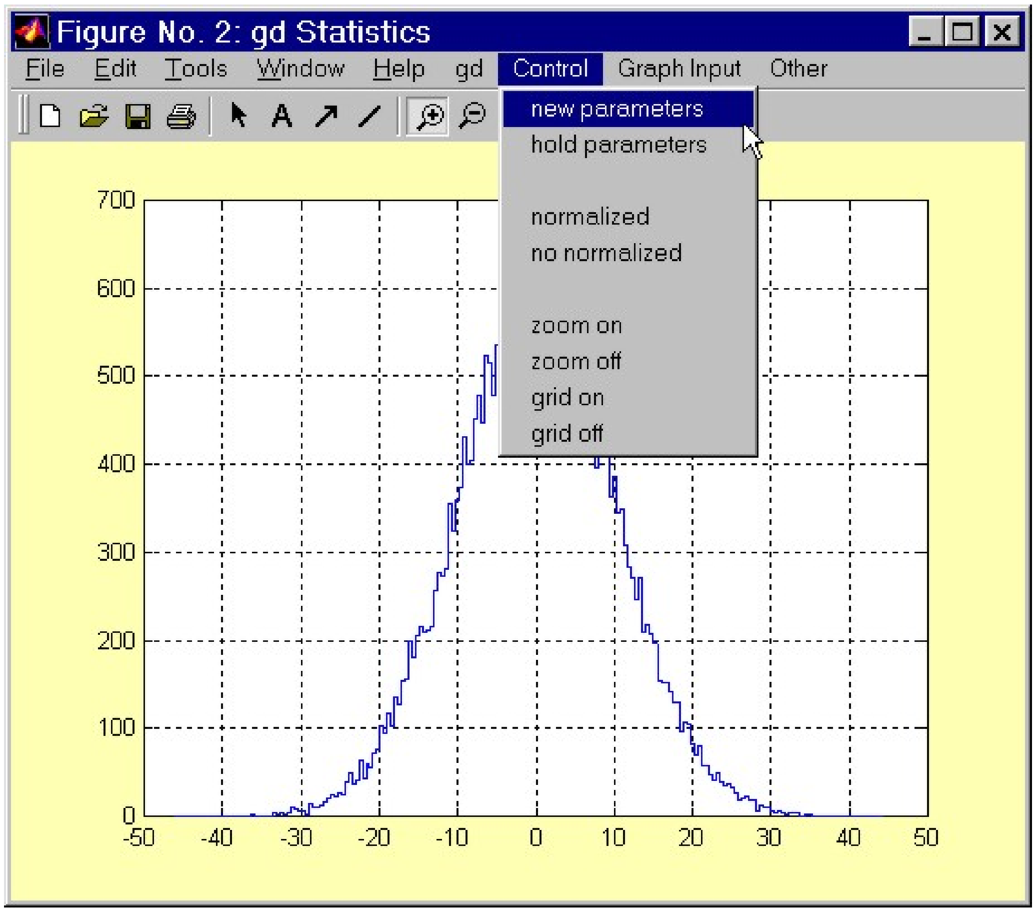
allows some interesting computations; the result appears in a pop-up window.



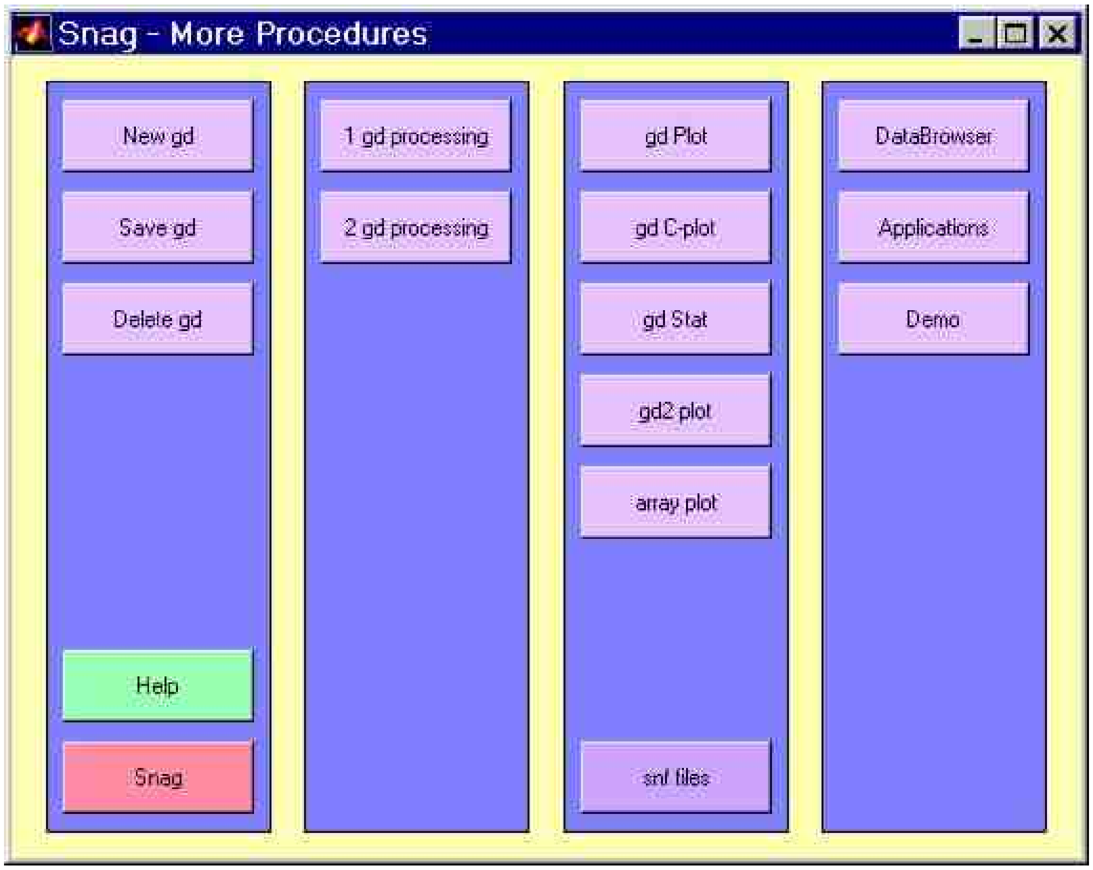
The **gd C-plot** button opens a similar facility to plot complex **gd**s



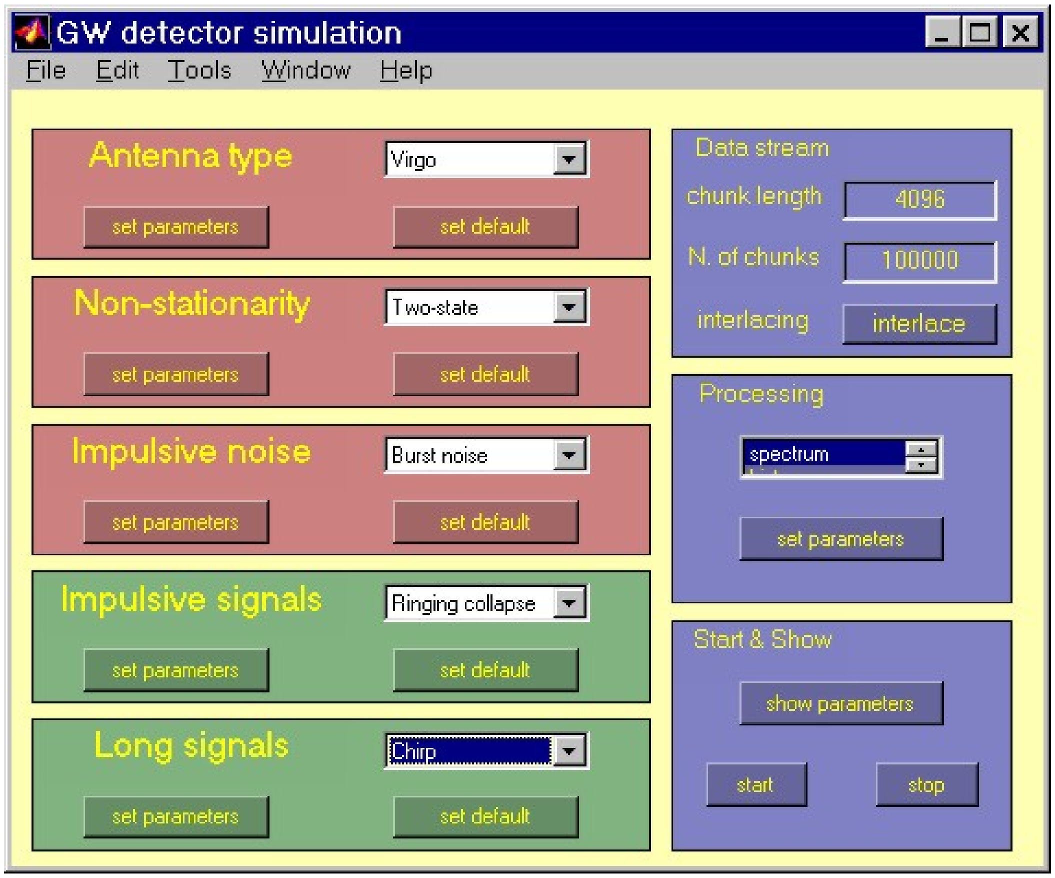
The **gd Stat** button opens an histogramming facility



There is also a menu **More**, that gives a different interface with more functions:



The **Applications** button opens other GUI applications, as **DataBrowser** and **GWSim**


GW detector simulation

Antenna type: Virgo

Non-stationarity: Two-state

Impulsive noise: Burst noise

Impulsive signals: Ringing collapse

Long signals: Chirp

Data stream

chunk length: 4096

N. of chunks: 100000

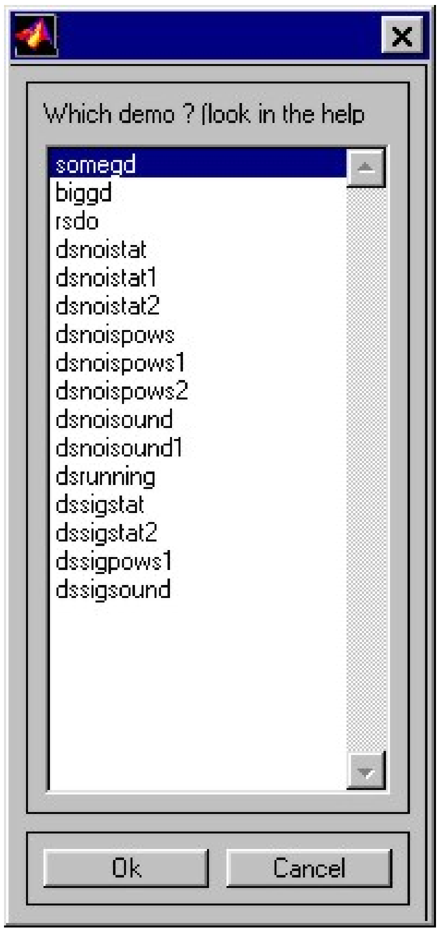
interlacing: interlace

Processing: spectrum

Start & Show

(in development).

The **Demo** button opens a list dialog



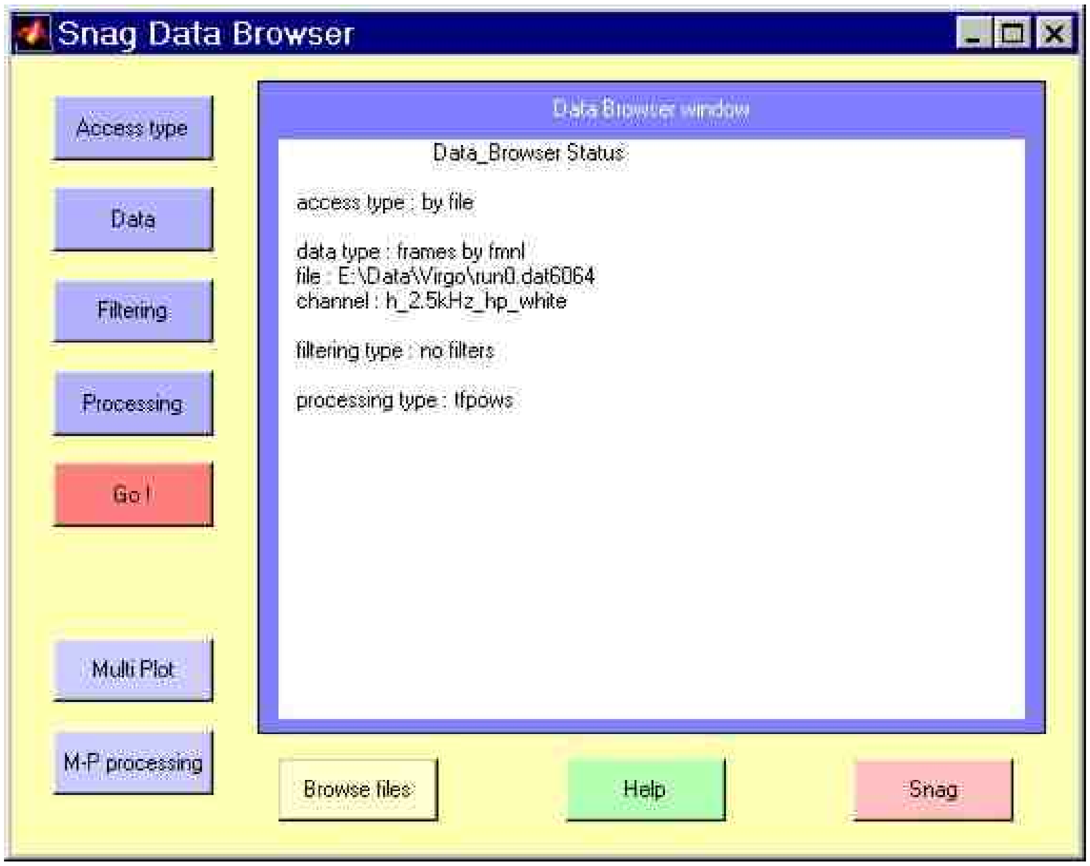
Which demo ? (look in the help

somegd
biggd
rsdo
dsnoistat
dsnoistat1
dsnoistat2
dsnoispows
dsnoispows1
dsnoispows2
dsnoisound
dsnoisound1
dsrunning
dssigstat
dssigstat2
dssigpows1
dssigsound

with some easy demo, useful to understand how to do easy programs using *Snag* .

data_browser

DataBrowser is a *Snag* application to access and simulate gravitational antennas data. It is started by typing **data_browser** (or just **db**, if the alias is activated). This opens a window



Data_Browser Status

access type : by file

data type : frames by fmnl
file : E:\Data\Virgo\run0.dat6064
channel : h_2.5kHz_hp_white

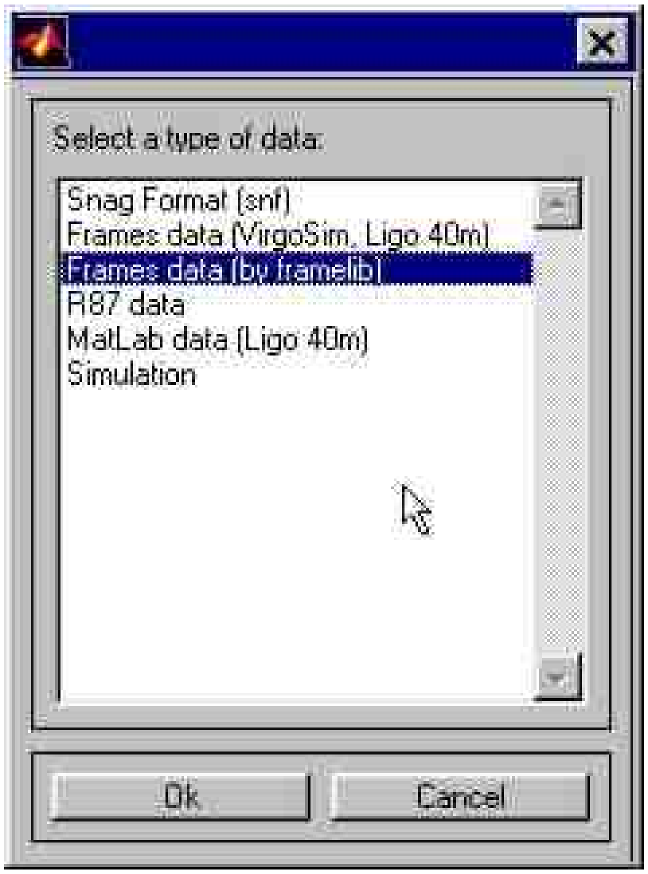
filtering type : no filters

processing type : tfpows

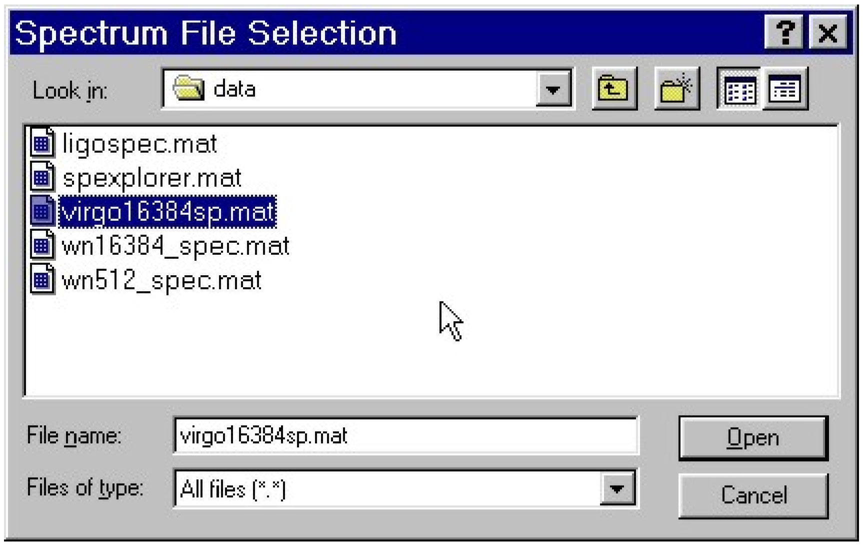
with a text window and some buttons.
The text window shows the "status" of the *DataBrowser* , as is due to the default and user's settings.
The first four buttons allow selections.

The **Access type** button allows the choice between the access the data by the name of the file or by the time. In the case of data simulation this choice make no sense.

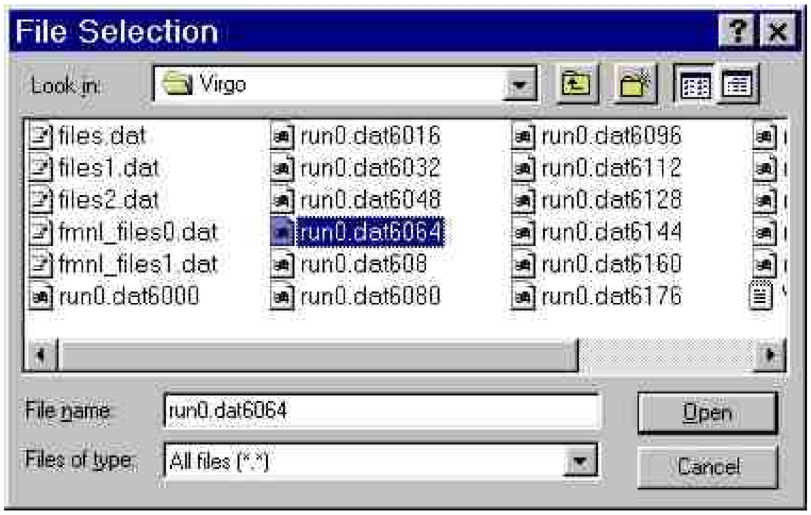
The **Data** button opens successively two or three windows. The first is



with a selection on some different types of data (possibly in different formats) or simulation. Choosing **Simulation**, a new window is opened



It shows a choice of some files containing **gd**s with the spectra of the data to be simulated. With a little knowledge of Matlab, more files can be added.
If, instead of **Simulation**, one chooses a type of data, two other windows will open:


File Selection

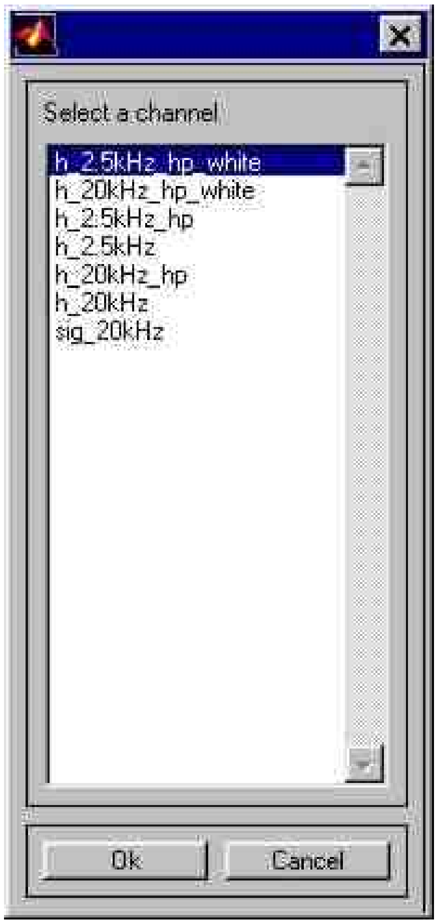
Look in: Virgo

files.dat
files1.dat
files2.dat
fmnl_files0.dat
fmnl_files1.dat
run0.dat6000
run0.dat6016
run0.dat6032
run0.dat6048
run0.dat6064
run0.dat608
run0.dat6080
run0.dat6096
run0.dat6112
run0.dat6128
run0.dat6144
run0.dat6160
run0.dat6176

File name: run0.dat6064

Files of type: All files (*.*)

that asks to select a file, and

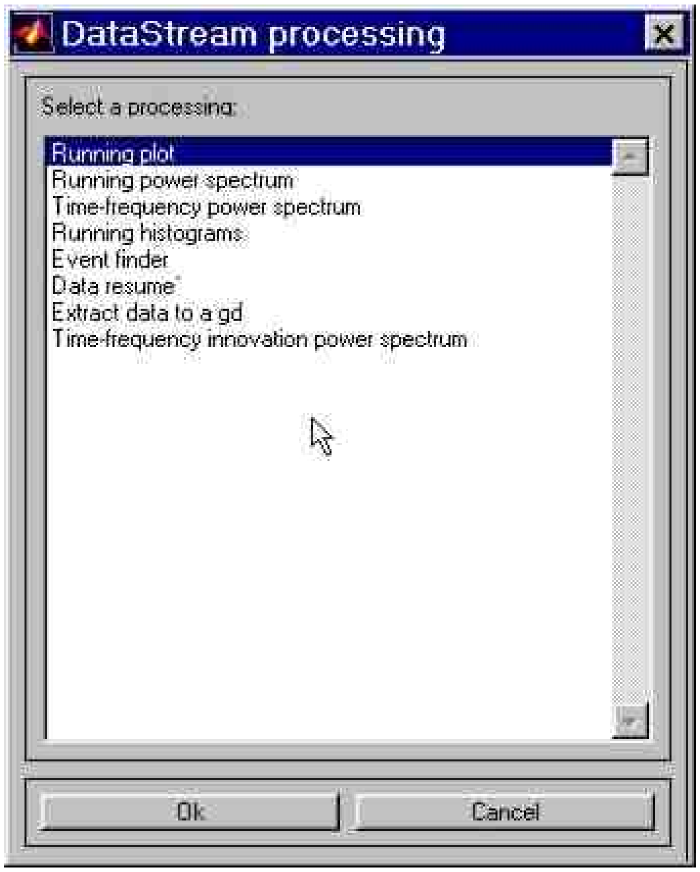

Select a channel

h_2.5kHz_hp_white
h_20kHz_hp_white
h_2.5kHz_hp
h_2.5kHz
h_20kHz_hp
h_20kHz
sig_20kHz

that asks to select a channel. To access the data correctly, the file **snag_local_symbol.m ,** that in the distribution is in the folder **/snag/local/,** must be updated and relocated in a different path. It contains the addresses of the data.

The **Filtering** button is used to chose a type of filtering.

The **Processing** button allows the choice of some processing on the data

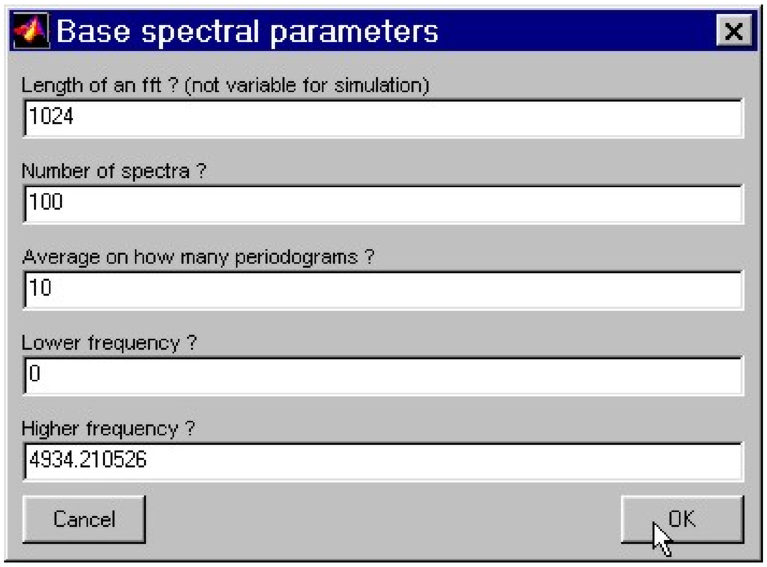


DataStream processing

Select a processing:

Running plot
Running power spectrum
Time-frequency power spectrum
Running histograms
Event finder
Data resume
Extract data to a gd
Time-frequency innovation power spectrum

The **Go !** button starts the processing, eventually after opening other parameter choice window (depending on the processing choice). For example, in the case of the time-frequency spectrum choice,


Base spectral parameters

Length of an fft ? (not variable for simulation)

1024

Number of spectra ?

100

Average on how many periodograms ?

10

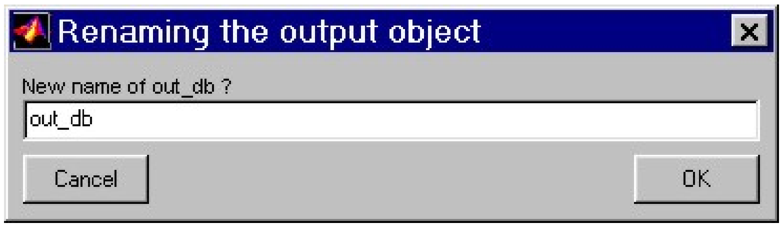
Lower frequency ?

0

Higher frequency ?

4934.210526

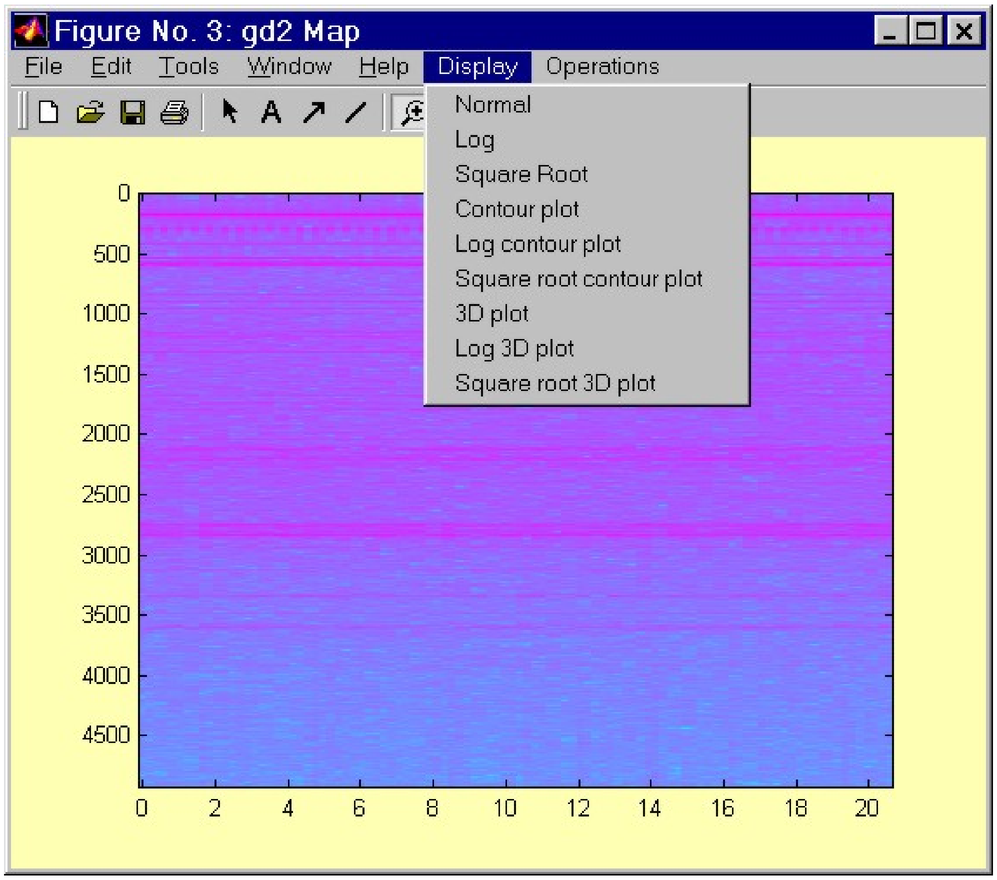
asks for the choice of the spectrum parameters. Then the spectrum is computed on the chosen data and at last dialog window is displayed


Renaming the output object

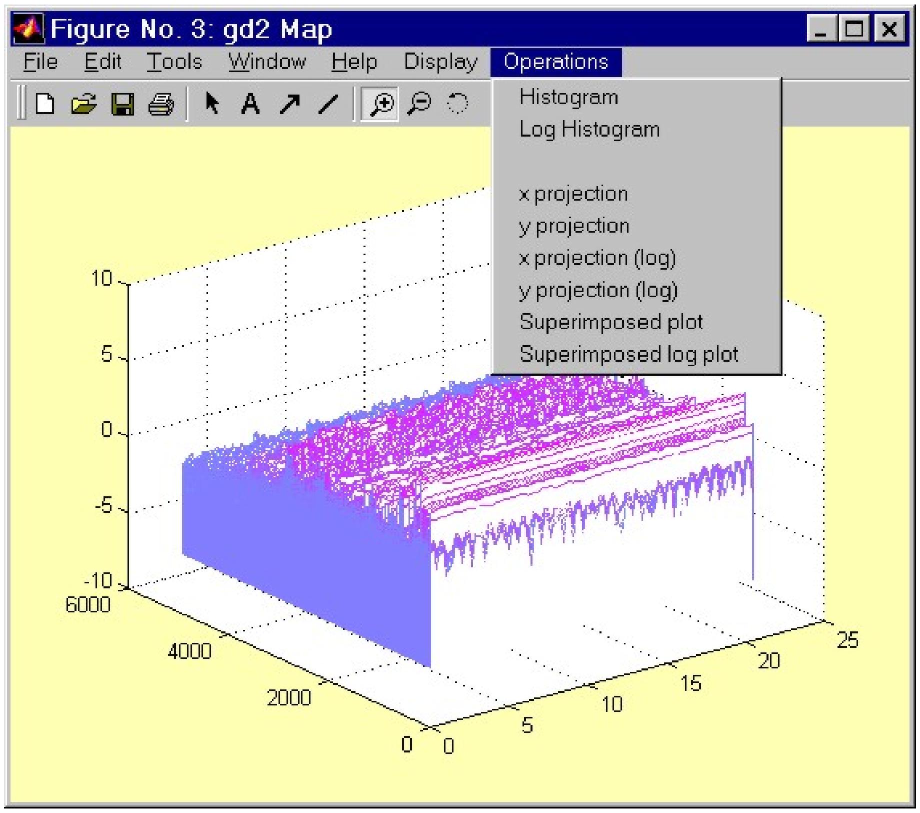
New name of out_db ?

out_db

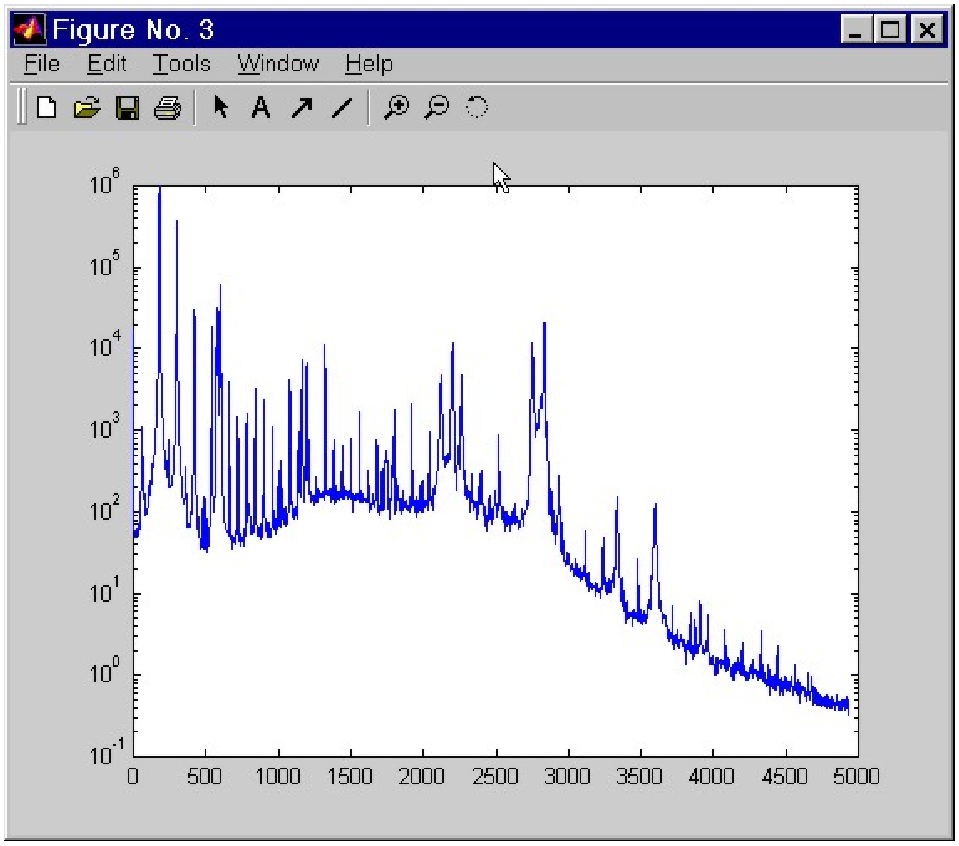
asking for the name to be given to the output object (in this case a **gd2** containing the time-frequency spectrum; this is useful if you want further to analyse or use these data) and a **gd2_map** window appears a map of the time-frequency spectrum, with the time in seconds on the abscissa and the frequency in hertz on the ordinates,



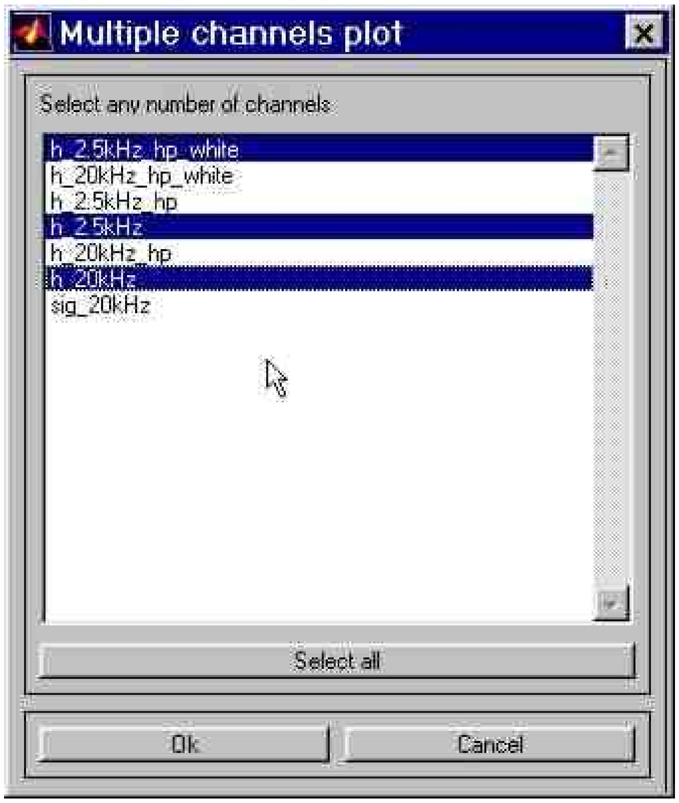
The **gd2_map** is a plotting facility for the **gd2**s, (**gd** with two dimensions, as a time-frequency spectrum). There are two user menus, **Display** and **Operations,** the first one with different types of plotting choices, the second one



with different types of operations allowed. For example, choosing **y projection (log),** appears

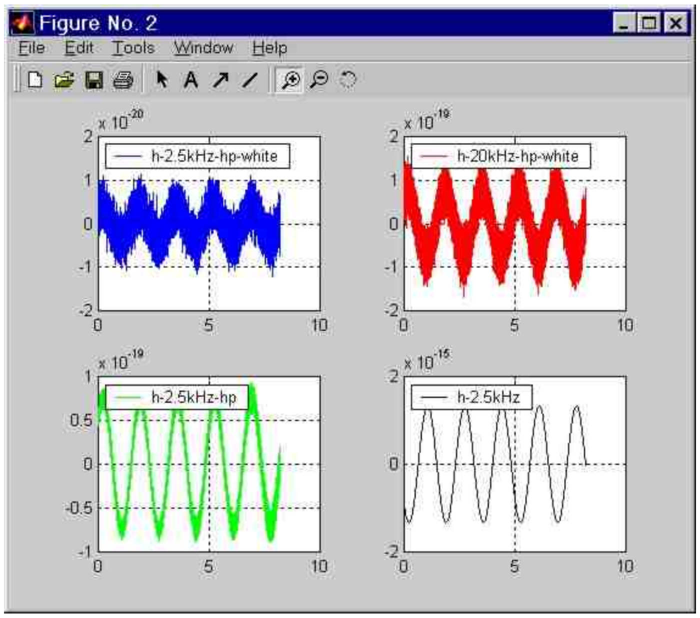


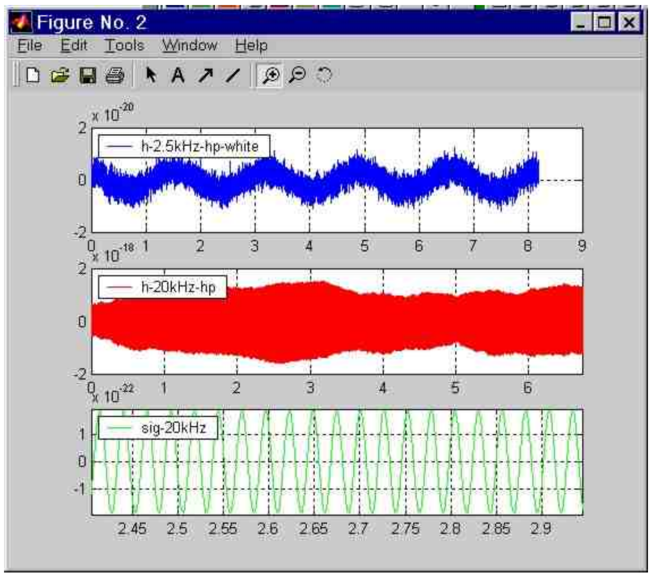
The access to the data (in the case of the frames) can be done also by the Multi Plot button. In this case, after having chosen a particular file, we can access and visualize simultaneously the data of more channels. In the following list dialog



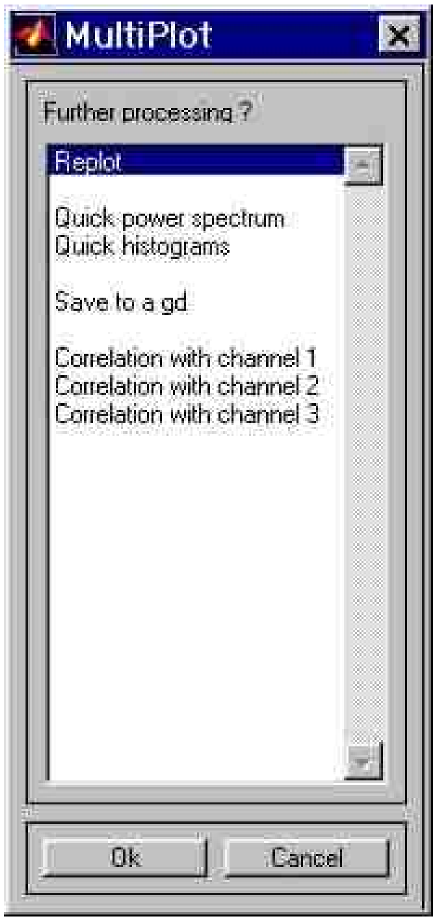
we can choose one or more channels to be visualized. Then we chose one of the four types of visualization:

- a single window with a single plot for all the channels
- the same as above, but with normalized values for all the channels
- a single figure with many distinct plots (see figures below)
- a window with a plot for each channel

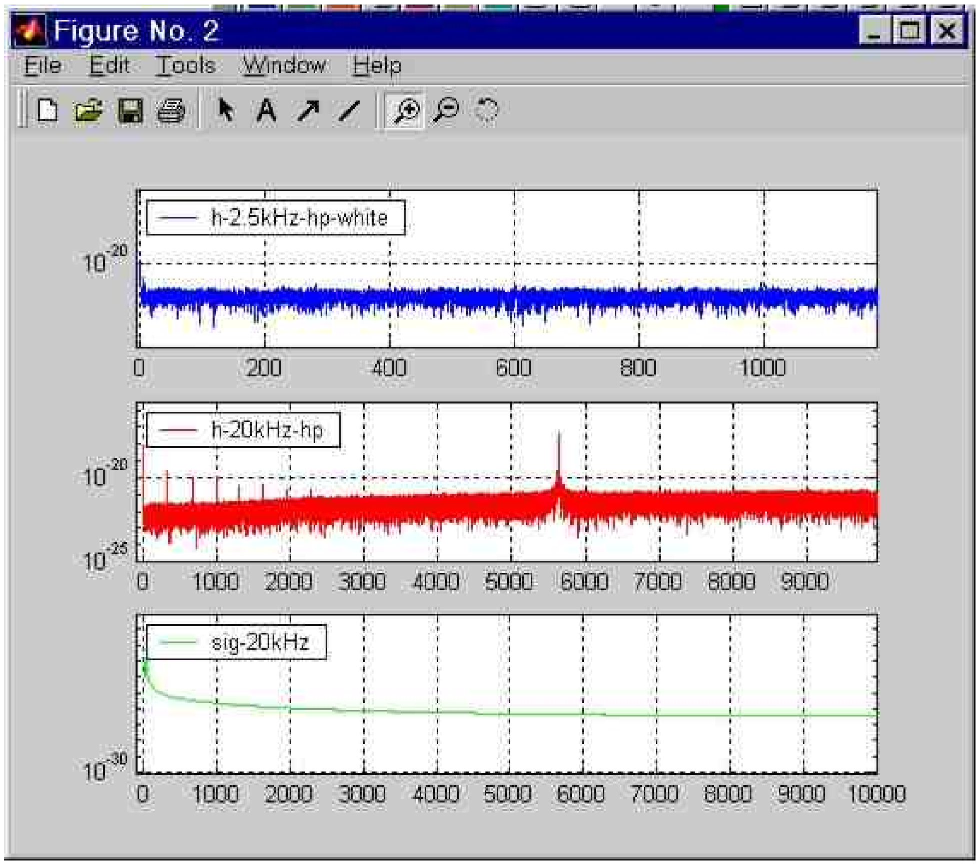




All plots can be zoomed individually. There is also the possibility to do some analysis on the plotted data; this can be accomplished by the M-P Processing button, with the following input list



Choosing one item means the application of the processing on all the channels; for example, choosing the "Quick spectrum", we have the spectra of all the channels:



The **Help** and **Snag** buttons are self-explaining.

Part II - Snag Data Analysis

Here a (small) collection of problems solved with snag is presented.

Basic data operation

Basic data analysis

Filtering

Event analysis

Part III - Snag main projects

Gravitational-Wave projects

Didactical projects

Appendix

Why Snag

Snag comes out as a "remake" (not a "porting") of an homonymous software developed from 1985 in the Rome gravitational wave research group. It was written in Fortran, under OpenVMS.
With the decay of the OpenVMS platform the necessity to port that software on other platforms came out.
Many possibilities were considered and some attempts were undertaken, in the years 1994-1998. Among the others:
- Snag_Dos, a porting to Dos done by a group of students by the PowerFortran compiler, a big work, but with many limitations due to the poor operative system.
- Snag_QW, a porting to Windows 95/NT, done with the Digital Fortran compiler, using the QuickWin library. A better result and with much less work. But it uses only partially the facilities of the Window environment and it remains a terminal mode application.
- SnagBCpp, an attempt to do a porting using Visual Basic for the GUI and Visual Basic and C++ for the operations.
- Some attempts using Visual C++ and Borland C++Builder.

The problems with these attempts were the time needed for the development and the practical impossibility to develop a really portable "professional" GUI application.
The problem was solved excellently by using MatLab, an "environment" used by many gravitational groups before us (for example the Louisiana group and the Potsdam group).
Here are the pros and cons of Matlab. The pros are of two types:
A - Matlab is a good environment:
- very easy high level programming language, to easy use the environment and develop applications, with good debugging facilities.
- a very large amount of well designed and tested mathematical functions, particularly for signal processing. This number is even greater with the numerous "toolboxes" (Signal Processing, Statistics, Wavelet, System Identification, Higher Order Spectral Analysis, etc.).
- many advanced graphic functions, easy to use and to enhance
- there are compilers that port the Matlab m-files to C or C++ programs

B - Matlab is a standard:
- very high number of users, in the industries and in the universities. Matlab has 400000 users, with more than 2000 universities; it is teached in many Engineering and Science departments
- there is a high number of "free" and commercial applications
- there are many software houses that do products to enhance the characteristics of Matlab, as Mathtools, or give compatible products, as MathViews

- there are free products that are "similar", as Scilab (http://www-rocq.inria.fr/scilab/) and GNU Octave (http://www.che.wisc.edu/octave/).
The main cons are:
A - Matlab is (relatively) expensive (in Italy the academic price is ~ 700 euros for the basic package)
B - It is not the best choice for non-interactive processing and when very high performances are needed.