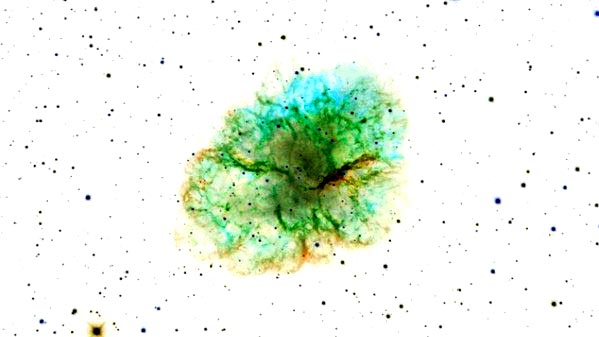
**BSD Manual**

**Band Sample Data Collection**

**Rome Virgo Group**

****

Last upgrade: 2023-05-19

Table of contents

[Table of contents 3](#_Toc118980978)

[Introduction 5](#_Toc118980979)

[BSD format 7](#_Toc118980980)

[The **tfstr** structure 9](#_Toc118980981)

[BSD folder 11](#_Toc118980982)

[BSD functions 13](#_Toc118980983)

[BSD creation 13](#_Toc118980984)

[Service functions 15](#_Toc118980985)

[**BSD Lego operations** 22](#_Toc118980986)

[**Sub-table extraction** 24](#_Toc118980987)

[**Choice of the procedure** 25](#_Toc118980988)

[**Cut and paste procedures** 26](#_Toc118980989)

[**BSD EquiLego operations (2021)** 27](#_Toc118980990)

[BSD basic 28](#_Toc118980991)

[BSD decoration 31](#_Toc118980992)

[BSD collection access 35](#_Toc118980993)

[BSD PSS 37](#_Toc118980994)

[**Work structures** 37](#_Toc118980995)

[**Functions** 38](#_Toc118980996)

[**BSD other searchs** 40](#_Toc118980997)

[BSD time events 41](#_Toc118980998)

[**BSD noncohe** 42](#_Toc118980999)

[**Typically batch procedures and reanalysis** 42](#_Toc118981000)

[sid\_sweep 42](#_Toc118981001)

[**Typically interactive procedures** 47](#_Toc118981002)

[sid\_sweep\_tf 47](#_Toc118981003)

[sid\_sweep\_ref 49](#_Toc118981004)

[sid\_sweep\_ref\_multisband 50](#_Toc118981005)

[sid\_sweep\_ref\_multidirec 50](#_Toc118981006)

[sid\_sweep\_multi 51](#_Toc118981007)

[bsd\_noncohe\_full 52](#_Toc118981008)

[Sidereal pattern analysis 53](#_Toc118981009)

[Other 55](#_Toc118981010)

[Service 56](#_Toc118981011)

[NC procedures 58](#_Toc118981012)

[**Basic engine** 58](#_Toc118981013)

[First level analysis 62](#_Toc118981014)

[Batch functions 64](#_Toc118981015)

[Service functions 66](#_Toc118981016)

[Data collection analysis 67](#_Toc118981017)

[BSD GUIs 87](#_Toc118981018)

[BSD procedures 88](#_Toc118981019)

[BSD analysis procedures 91](#_Toc118981020)

[Non-bsd useful functions 93](#_Toc118981021)

[Service 93](#_Toc118981022)

[VFS (Varying Frequency Signal) 95](#_Toc118981023)

[MC Filter 98](#_Toc118981024)

[Example scripts 99](#_Toc118981025)

[Extracting events 99](#_Toc118981026)

[Extracting narrow bands 102](#_Toc118981027)

[Correction of Doppler and spin-down 103](#_Toc118981028)

[Software injections 105](#_Toc118981029)

[Compute 5-vec 107](#_Toc118981030)

[equi\_example 109](#_Toc118981031)

[Code Blocks (“BSD snippets”) 111](#_Toc118981032)

[**General job program** 115](#_Toc118981033)

[Appendix 116](#_Toc118981034)

[Debug & check 116](#_Toc118981035)

[Index 117](#_Toc118981036)

Introduction

The **gd** is the basic snag class or an object of this class.

A **bsd** is a type 1 gd that contains complex sampled data with information on the gravitational wave signal in a band.

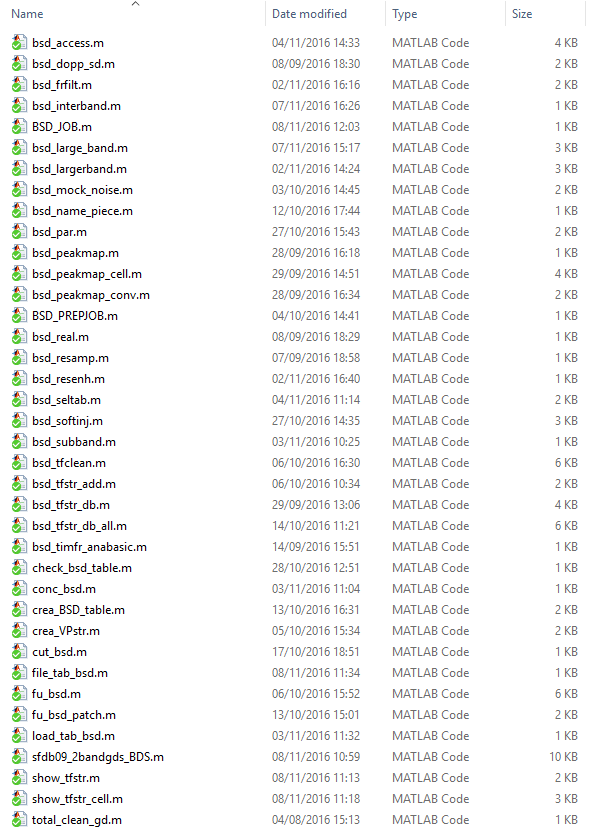
The bsd have a particular cont structure.

There are basic or “**primary**” bsd that are stored and collected in a particular folder structure, the BSD collection.

Starting from a BSD collection, one can construct other “**secondary**” bsd.

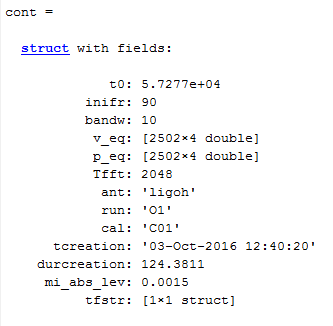
Another document on bsd is the **PSS\_UG\_III.docx** that describes in more details the use of the bsd for the PSS search.

At 8-Nov-2016:



BSD format

A bsd is a standard gd with peculiar **cont** structure (primary bsd):



**t0** is the mjd of the first sample (the ini of the gd is 0 and the dx is the sampling time in seconds.

**inifr** is the lower frequency of the band in Hz.

**bandw** is the bandwidth (in Hz)

**v\_eq** is the detector velocity components in unit of c, with the time in s since the beginning of data

**p\_eq** is the detector position components in a SSB system in unit light seconds, with the time in s since the beginning of data

**Tfft** is the duration of the original SFDB ffts (in practice almost useless)

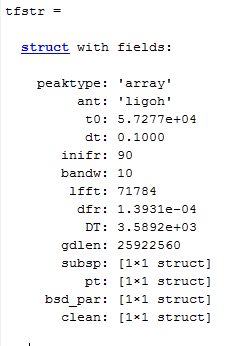
**ant**, **run** and **cal** are the antenna, run and calibration

**tcreation** and **durcreation** are the creation time and duration

**mi\_abs\_lev** the standard amplitude computed by logarithmize (the cleaning threshold is at 10\*mi\_abs\_lev)

**tfstr** time-frequency “decoration” structure.

The time-frequency “decoration” structure is:



To extract the tfstr, use the function tfstr\_gd(*decorated bsd*):

function tfstr=tfstr\_gd(gin)

% extracts tfstr structure

% Version 2.0 - September 2016

In secondary bsd can be an **oper** structure that contains information and parameters used to generate it.

If the starting bsd was secondary, the oper structure contains inside the original oper structure. And so on, recursively.

A **full-band bsd** is a particular type of bsd, that has the entire (Nyquist) band. In this case some parameters can be missing. It has the half sampling frequency of the real data.

## The **tfstr** structure

The elements of the structure are:

* **peaktype** array or cell
* **ant,t0,dt,inifr,bandw** as in cont
* **lfft** used fft length (in samples
* **dfr** frequency bin
* **DT** =lfft\*dt/2 time between two ffts
* **gdlen**  number of samples in the original gd
* **subsp**  short spectrum
  + **Nsstim**
  + nss
  + lss
  + init
  + typ
  + hdens
  + wn
* **pt**  peak table structure
  + **thr**
  + tpeaks
  + npeaks
  + ntotpeaks
  + peaks
  + ntotpeaks
  + ok
* **bsd\_par** bsd parameters
  + **frmax**
  + tfft0
  + lfft0
  + dfr0
  + Ndop
  + sky
    - x
    - b
    - index
    - nlon
    - nskypoint
    - nbeta
    - x2
    - maxbeta
    - minbeta
* **clean** cleaning structure
  + **nt**
  + **nfr**
  + **NF**
  + **NT**
  + **DF**
  + **DT**
  + **PHT**
  + **mpers**
  + **persist**
  + **tfhist**
  + **htfhist**
  + **xhtfhist**
  + perscutfr
  + tfcut
  + **persist0**
  + **tfhist0**
  + **htfhist0**
  + **xhtfhist0**

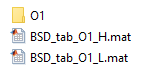
BSD folder

The BSD folder contains one folder for each run and all the BSD tables.

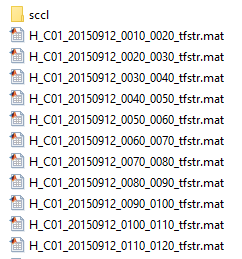
Any run folder contains a folder for each antenna.



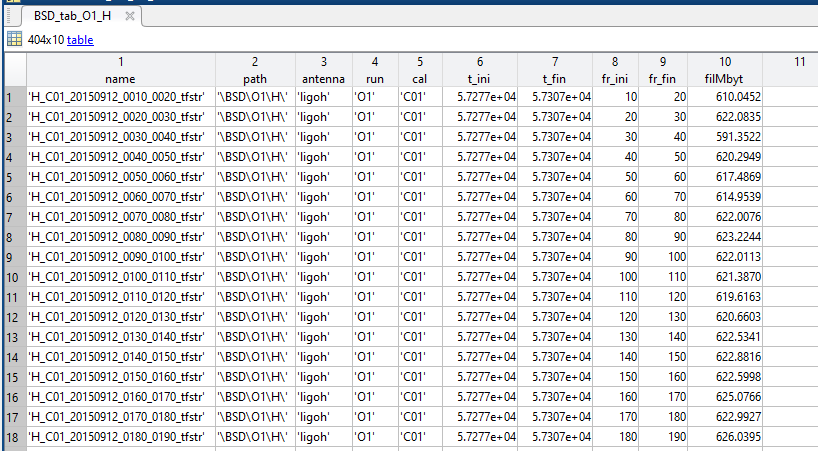
Content of BSD folder:



Content of O1 H folder:



BSD table for O1 Hanford:



BSD functions

There are many functions operating on BSDs. Here is a list of the most basic. There are also folders with procedures, examples, check&debug, and so on.

There is also a collection of **global variables** used to modify the behavior of some functions:

* bsd\_glob\_noplot
* bsd\_glob\_level
* bsd\_glob\_stack
* bsd\_glob\_stacklev
* bsd\_glob\_nozeroing

**Remember to reset (set to 0) global variables at the end of use. The reset\_bsd\_glob() function may be used.**

BSD creation

function [g, y, itimes, headers]=…

sfdb09\_2bandgds\_BDS(pialist,foldout,frmin,frmax,BSD\_BAND,nfft,run,sim)

%SFDB09\_2BANDGDS creates BSD gds

% fseek

% out=sfdb09\_2bandgds(pialist,foldout,frmin,frmax)

%

% pialist input file list

% foldout output gds folder

% frmin,frmax operation range (depends on the number of gds to manage simultaneously)

% (for one gd 210 MB are needed)

% calibration sting e.g. 'C00' or 'C01' indicates the calibration version

% sim =1 usa solo la sinusoide

% diverso da 2 e 1 usa i dati veri altrimenti una sinusoide al posto dei dati

% =2 usa dati+sinusoide

% struct, ex.:

% sim.type = 3 implemented in v10

% .VP VP array

% .fr frequency

%

% sim.type = 4 not implemented

% .VP VP array

% .fr central frequency

% .fivev 5-vec

% if window=0 window the data

% HEADER

% piahead.einstein=fread(fid,1,'float32'); %

%

% piahead.detector=fread(fid,1,'int32'); % detector (0,1)

% piahead.tsamplu=fread(fid,1,'double'); % original sampling time

% piahead.typ=fread(fid,1,'int32'); % interlacing (0, 2)

%

% piahead.wink=fread(fid,1,'int32'); % window type

% piahead.nsamples=fread(fid,1,'int32'); % number of samples of fft

% piahead.tbase=fread(fid,1,'double'); % length of fft in s

% piahead.deltanu=fread(fid,1,'double'); % fft frequency bin

% piahead.firstfrind=fread(fid,1,'int32'); % first index (e.g.: 0)

% piahead.frinit=fread(fid,1,'double'); % initial frequency

% piahead.normd=fread(fid,1,'float32'); % factor |fft|^2 -> pow spect

% piahead.normw=fread(fid,1,'float32'); % factor window (to be multiplied for normd)

%

% piahead.red=fread(fid,1,'int32'); % reduction factor (very short spectrum)

%

% !!! NO SQRT(2) CORRECTION FOR SFDB09 CREATED AFTER 15 APRIL 2010

% !!! for sfdb09 files created before, use sfdb09\_band2sbl\_rad2

% Snag Version 3.0 - April 2016

Service functions

function bsd\_glob\_set=reset\_bsd\_glob()

% resets all global variables

%

global bsd\_glob\_noplot bsd\_glob\_level

bsd\_glob\_noplot=0

bsd\_glob\_level=0

bsd\_glob\_set.noplot=bsd\_glob\_noplot;

bsd\_glob\_set.level=bsd\_glob\_level;

function gdin=total\_clean\_gd(gdin)

%

% 1.sel\_data\_v3 --> selects science segments data using sciseglist

% 2.logarithmize --> evaluates the median abs level to be used in 3. (\*10 typical threshold)

% 3.complex\_clip --> applies cleaning usign the threshod compute in 2.

% 4.add\_zeros\_gd\_ini --> add zeros at the beginning of the gd, makes the gd

% start at h 0 of the relative day of the first data

%

% gdin gd to be cleanded

% sciseglist science segmennt list (different format of files pay attention) (e.g '/storage/users/piccinni/O1\_H1\_segments\_science\_full.txt')

%

%%O.J.Piccinni july 2016

tic

[~,mi]=logarithmize(gdin)

gdin=complex\_clip(gdin,10\*mi);

% gdin=add\_zeros\_gd\_ini(gdin);

cont=cont\_gd(gdin);

cont.tcreation=datestr(now);

cont.durcreation=toc;

cont.mi\_abs\_lev=mi;

gdin=edit\_gd(gdin,'cont',cont);

function [VPstr,contout]=crea\_VPstr(ant,timdop,doptabs,ingd)

% vel,pos,einst for some time samples

%

% ant antenna structure or name

% timdop times for doppler data [start (mjd) samp.t (s) n] ! def values for 0

% doptabs doptabs structure (a field each antenna,

% columns are (mjd posx posy posz velx vely velz einst)

% ingd imperfect gd or reuested

%

% VPstr used in interp\_VP and other bsd software

% Version 2.0 - October 2016

function name\_piece=bsd\_name\_piece(name,prot)

% decodes bsd name pieces

%

% name\_pieces=bsd\_name\_pieces(name)

%

% Version 2.0 - October 2016

function BSD\_tab=crea\_BSD\_table(list,tabname)

% creates a BSD table from a list of files

%

% list file list with path (ex.: 'list\_O1\_L\_tfstr.txt')

% tabname table name (ex.: 'BSD\_O1\_L')

% Version 2.0 - October 2016

function [bsd\_o,i1,i2]=cut\_bsd(bsd\_i,tt) **% OPER structure**

% cuts a bsd

%

% bsd\_i input bsd

% tt [tin tfi] time interval (mjd or seconds from beginning)

% Version 2.0 - October 2016

function [bsd\_o,ini2]=conc\_bsd(bsd\_i1,bsd\_i2)

% concatenates two bsd that should be compatible (same band, same sampling time)

%

% bsd\_i1 first input bsd

% bsd\_i2 first input bsd

% Version 2.0 - October 2016

function out=check\_bsd\_table(tab)

% checks a bsd table

% Version 2.0 - October 2016

function tab\_out=bsd\_seltab(tab,ii,varargin)

% bsd table selection

%

% tab master table

% ii selection indices

% varargin couples of variables (item, values)

% Version 2.0 - November 2016

function files=file\_tab\_bsd(addr,tab)

% adds path and extention to bsds from a table

%

% files=file\_tab\_bsd(addr,tab)

%

% addr BSD path (without final dirsep)

% tab bsd table

% Version 2.0 - October 2016

function [bsd, name]=load\_tab\_bsd(addr,tab,k,modif)

% loads a bsd from a table

%

% [bsd, name]=load\_tab\_bsd(addr,tab,k)

%

% addr BSD path (without final dirsep)

% tab bsd table

% k file index

% modif modification structure

% Snag Version 2.0 - October 2016

function tab\_out=bsd\_extr\_subtab(tab, tim,fr)

% extracts a sub table from a bsd table

%

% tab input table

% tim any time in the file (0 no choice)

% fr any frequency in the file (0 no choice)

% Snag Version 2.0 - May 2017

function par=bsd\_par(in,parin) %% CHECK !!

% parameters for BSDs

%

% in input bsd

% parin (if present) input parameter structure

% Version 2.0 - September 2016

function out=bsd\_subband(in,band,st)  **% OPER structure**

% extract band (output signal is complex)

% [similar to extr\_band]

%

% out=bsd\_subband(in,band,st)

%

% in input gd (type 1)

% band [min max] frequency (aliased)

% st sampling time (if present; otherwise st=1/band)

%

% ATTENTION ! the samples start since the 0 hour of the first day of the run

% so, the first sample of the file can be advanced: in this

% case the first sample is a 0

% Version 2.0 - September 2016

function [out,st,band,pars]=bsd\_reshape(in,tab,bandin,t00)

% reshapes a bsd in order to concatenate sub-bands

% corrects starting sample and starting frequency and length

%

% [out,band]=bsd\_reshape(in,bandin,t00)

%

% in input bsd

% tab table of the bsds to be concatenated

% bandin requested band [frin frfin]

% t00 requested basic time (optional - def 0 hour of run starting day)

%

% out reshaped bsd

% dfr frequency quantum

% band true band

% Snag Version 2.0 - November 2016

function [bsd\_sb,pars]=bsd\_concsubband(addr,BSD\_tab\_out,bandin,modif)

% sub-band of concatenated bsds

%

% [bsd\_sb,pars]=bsd\_concsubband(addr,BSD\_tab\_out,bandin,modif)

%

% addr path that contains BSD data master directory without the final dirsep

% BSD\_tab\_out operative file table

% bandin requested band

% modif modif structure (or cell array) for bsd\_acc\_modif

% Snag Version 2.0 - November 2016

function [zhole, holes, dholes, thole]=bsd\_holes(in)

% identifies holes in a bsd

%

% in input bsd

%

% holes a vector with odd elements index of starting data and even elements index of starting hole

% thole (n 2) start and stop mjd of holes

% Snag Version 2.0 - November 2016

function out=bsd\_zerointerv(in,zint)

% zeroes intervals (as holes...) of a bds

%

% out=bsd\_zerointerv(in,zint)

%

% in input bsd

% zint (n,2) couple of indices (start stop)

% Snag Version 2.0 - November 2016

function out=bsd\_zeroholes(in1,in2)

% forces to zero the holes

%

% in1 bsd to be zeroed

% in2 bsd with the holes

% Snag Version 2.0 - November 2016

function out=bsd\_dayplot(bsd,typ)

% plots a bsd with better abscissa

%

% out=bsd\_dayplot(bsd,typ)

%

% bsd input bsd

% typ type of abscissa start:

% 1 day (default)

% 2 week

% 3 month

% 4 year

% Snag Version 2.0 - January 2017

function yn=is\_bsd(in)

% type of bsd

%

% yn=is\_bsd(in)

%

% in gd

%

% yn = 0 no, simple gd

% = 1 basic bsd

% = 2 complete primary

% = 3 complete secondary

% = 4 complete oper present

% = 5 complete decorated

% Snag Version 2.0 - January 2017

function [inifr,bandw,dt,iok]=bsd\_orthoband(inifrs,bandws,dt0)

% computes the smallest "fine" band for bsds

%

% inifrs,bandws signal band

% Snag Version 2.0 - February 2017

function lines=ligo\_readlines(fil)

% reads lines ligo files (in csv format)

%

% fil lines Ligo file (in csv format)

%

% typical file: O2H1lines.csv

% Column 1 - frequency spacing (Hz) of comb (or frequency of single line)

% Column 2 - comb type (0 - singlet, 1 - comb with fixed width, 2 - comb with scaling width)

% Column 3 - frequency offset of 1st visible harmonic (Hz)

% Column 4 - index of first visible harmonic

% Column 5 - index of last visible harmonic

% Column 6 - width of left band (Hz)

% Column 7 - width of right band (Hz)

%

% For fixed-width combs, veto the band:

% [offset+index\*spacing-leftwidth, offset+index\*spacing+rightwidth]

% For scaling-width combs, veto the band:

% [offset+index\*spacing-index\*leftwidth, offset+index\*spacing+index\*rightwidth]

% Snag Version 2.0 - November 2018

function pulsar=read\_pulsar\_Ligo\_file(plist,kpuls,gpsref)

% Creates a cw source structure from Ligo list

%

% plist file name (csv type)

% kpuls number of the pulsar

% gpsref gps reference time (default from the name of the csv file)

% Snag Version 2.0 - November 2018

## **BSD Lego operations**

Here the basic composition rules and procedures for cut and paste BSDs are described. These are the basic operations:

**Time operation:**

1. interval extraction
2. appending periods

**Frequency operation:**

1. sub-band extraction
2. more bands synthesis

Often an **A** operation is connected with a **D** operation and can be performed in “large band” way (the starting frequency is the lowest frequency) and “larger band” way (the starting frequency is 0 and the original frequencies are reconstructed).

A **C** operation is often connected with a **B** operation.

A particular type of sub-band is an **ortho-band**. The full band of a BSD is the band



with Δt the sampling time. An ortho-band has a width with k integer and is limited by the frequencies (j-1)·B and j·B. Therefore, an ortho-band is defined by two integer number, j and k. The sampling time associated to an ortho-band is .

To guarantee the correct connection of data from different periods in a sub-band, it is important that:

* all the sub-bands for the different pieces are absolutely equal
* the sampling time of the

There are many ways to cut and paste BSDs. The basic problems are

* avoid discontinuities
* avoid artifacts
* operate with the least memory
* be fast
* be simple for users
* be simple for programmers

**To operate in the best way the ortho-band should have a number of bins that is a submultiple of the length of the fft used for the cut and paste operation. With this choice, the new sampling time is a multiple of the original sampling time.**

The cut-and-pasted data form a new bsd.

The procedure to build the new BSD is **bsd\_lego** (former bsd\_access). It is composed of 3 parts:

1. extract the sub-table of the necessary BSDs
2. define the procedure to use
3. apply the procedure

### **Sub-table extraction**

### **Choice of the procedure**

### **Cut and paste procedures**

## **BSD EquiLego operations (2021)**

A better way to concatenate BSDs for short bands is performed by the EquiLego procedures. With these procedures there is no limitation to ortho-bands: any band, contained inside one or two adjacent basic band can be extracted.

function [bsd\_out,BSD\_tab\_out,stpar]=bsd\_equilego(addr,tab,tim,freq,modif,modifpost)

% data access alternative to bsd\_lego

% good for any subband and more months

%

% limits:

% no modes (there are two automatic modes)

% out band could be any, but should be inside 2 adjacent basic BANDs

%

% [bsd\_out,BSD\_tab\_out,stpar]=bsd\_equilego(addr,tab,tim,freq,modif,modifpost)

%

% addr path that contains BSD data master directory without the final dirsep

% or structure with all the input parameters as fields

% tab BSD table (charged)

% tim [tini tfin] (mjd) ; if length(tim) = 1 or tini=tfin the whole run

% [tini tfin dt0] if the sampling time dt0 of the BSD collection is not 0.1 s

% freq [frini frfin] ; if length(freq) = 1 or frini=frfin the full band with frini (def ortho-band)

% (freq is the requested band, band will contain the effective band)

% modif (if present) modification structure: modifies accessed primary files

% e.g. adding signals or sources; see bsd\_acc\_modif

% [] no modification

% modifpost (if present) modification after the band extraction

Example:

[bsd\_out,BSD\_tab\_out,stpar]=bsd\_equilego('K:',BSD\_O3\_H,0,[108 109])

[bsd\_out,BSD\_tab\_out,stpar]=bsd\_equilego('K:',BSD\_O3\_H,0,[109 111])

uses

bsd\_out=bsd\_interband\_equilego(addr,tab,stpar,modif)

## **base\_orthobands**

function out=base\_orthobands(bandin,dt)

% basic orthobands info

%

% bandin requested band

% dt sampling time

%

% base orthobands

% k

% dk 1 2 3 4 5 6 7 8 9...

% 0 \* \* \* \* \* \* \* \* \* \* \* \*

% 1 \*

% 2 \* \*

% 3 \* \*

% 4 \* \* \*

% 5 \* \*

% 6 \* \* \* \*

% 7 \* \*

% 8 \* \* \* \*

% 9 \* \* \*

% 10 \* \* \* \*

% 11 \* \*

% 12 \* \* \* \* \* \*

%

% OBpar 1 bw, band width, in basic units (1,2,3,...)

% 2 bs, band start, in bw units (0,1,2,...)

% 3 sbd, sub-band divisor (1,2,3,...)

% 4 sbk, sub-band k (1,...,sbd)

%

% red = sbd/bw

% dtout = dt\*red

BSD basic

function out=bsd\_real(in)  **% OPER structure**

% conversion to real (doubling sampling frequency)

%

% out=bsd\_real(in)

% Snag Version 2.0 - September 2016

function out=bsd\_freqshift(in,dfr,tref)

% frequency shift for bsd. Shift is circular.

%

% in input bsd

% dfr frequency shift

% tref reference time (def 14 sep 2015)

% Snag Version 2.0 - April 2021

function out=bsd\_enlarge(in,enl)

% enlarge frequency domain

%

% in input bsd

% enl frequency domain enlargement (def 2)

% Snag Version 2.0 - April 2021

function out=bsd\_dopp\_sd(in,sour,sdpar) **% OPER structure**

% Doppler and spin-down correction

%

% out=bsd\_dopp\_sd(in,sour,sdpar)

%

% in input bsd

% sour source structure

% sdpar spin-down parameters, if different from source

% [df ddf] or df; if sd=0, no spin-down correction

% if absent, uses sour parameters

% Version 2.0 - September 2016

function gout=bsd\_mock\_noise(gin,kfact,nonst) **% OPER structure**

% bsd mock noise

%

% gin input bsd (to simulate)

% kfact reduction factor (integer)

% nonst (if present) non-stationary mask (with holes)

% Version 2.0 - September 2016

function tfstr=bsd\_tfclean(tfstr,anabasic)

% time-frequency cleaning

%

% tfstr=bsd\_tfclean(tfstr,anabasic)

%

% - tf filter

% - persistence

% - noise adaptive filter

%

% tfstr tf structure as created by bsd\_peakmap

% anabasic procedure control structure

% .pers\_thr persistence threshold (0 -> no threshold; typical 0.3)

% .tfh\_df time-frequency histogram df

% .tfh\_dt time-frequency histogram dt (days) def 0.5

% .tfh\_pht time-frequency histogram time phase (hours) def 8 local time

% .tfh\_thr time\_frequency threshold (0 -> no threshold)

% .noplot = 0 plot (default)

% Version 2.0 - September 2016

function [sbsd,fr,sig0]=bsd\_softinj(gin,sour,phs,A) **% OPER structure**

% bsd software injection

%

% sbsd=bsd\_softinj(gin,sour,phs,A)

%

% gin input bsd (or bsd data structure)

% sour source

% phs input phases in degrees (ph0, phbin, sidph, phase epoch) [4 components]

% A amplitude (if not present, use sour h)

% Version 2.0 - October 2016

function out=bsd\_largerband(list,tin,tfi,addr,outdt)

% synthetic signal from more BDSs

%

% out=bsd\_largerband(list,tin,tfi,outdt)

%

% list bsd table or file with bds files list (NO SPACES)

% (if a row of the list begins with \*, it is not considered)

% tin initial time (mjd)

% tfi final time

% addr (used in case of table) path without dirsep; otherwise ''

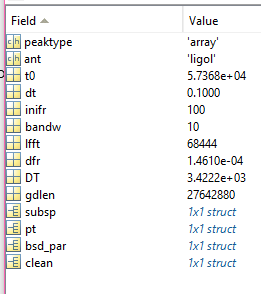
% outdt (if present) output file sampling time

% Version 2.0 - September 2016

function out=bsd\_resamp(in)

BSD decoration

The “decoration” of a BSD is the addition of more time-frequency information. This data are grouped in the tfstr structure added to the cont structure (see in the BSD format section)



It is created by

function out=bsd\_tfstr\_add(list,anabasic)

% adds tf structure to bsds

%

% list bsds file list (file should end with sccl)

% anabasic basic analysis structure (if absent, default is used)

% Version 2.0 - September 2016

or, if notch filter is applied,

function out=bsd\_tfstr\_addwithnotch(list,parnotch,anabasic)

% adds tf structure to bsds performing notch filtering

%

% list bsds file list (file should end with sccl)

% parnotch notch parameters (if absent or a void structure or not a structure, default values)

% .thr threshold on persistence

% .win window (def 1)

% .linel line list [n,2] (frmin frmax)

% anabasic basic analysis structure (if absent, default is used)

% Version 2.0 -November 2018

that uses

function out=bsd\_notch(in,parnotch)

% notches on bsds

%

% out=bsd\_notch(in,parnotch)

%

% in input bsd

% parnotch notch parameters (if absent or a void structure or not a structure, default values)

% .thr threshold on persistence

% .win window (def 1)

% .linel line list [n,2] (frmin frmax)

% Snag Version 2.0 - November 2018

**2019 Addition: holes info**

The function **bsd\_tfstr\_add** adds also holes info.

The function

function out=add\_zeros2tfstr(in,addr)

% adds zeros info to tfstr

% start from a non-BSD directory

%

% in input bsd or table

% addr address of the table (in the case of table, e.g. 'I:')

%

% out output bsd or number of saved bsd

Adds the holes info on pre-existing bsd files.

The holes info is in a two column arrays with the beginning and end time of holes, in seconds from the beginning.

Here is a list of the function used to create and analyze the tfstr data.

function [tfstr,out]=bsd\_timfr\_anabasic(in,anabasic)

% basic time-frequency analysis for bsd files

%

% [tfstr,out]=bsd\_timfr\_anabasic(in,anabasic)

%

% in input bsd

% anabasic analysis control structure

% .thr spectral peak threshold (default 2.5)

%

% tfstr analysis output structure

% out (if present) t-f enriched bsd

% Version 2.0 - September 2016

function out=show\_tfstr(in)

% shows features of a tfstr

%

% in bsd gd or tfstr

% Version 2.0 - September 2016

function out=show\_tfstr\_cell(in)

% shows features of a tfstr

%

% in bsd gd or tfstr

% Version 2.0 - September 2016

function tfstr=bsd\_peakmap(in,lfft,thr,typ)

% peakmap creation

%

% tfstr=bsd\_peakmap(in,lfft,thr,typ)

%

% in input bsd

% lfft length of the ffts (multiple of 4)

% thr normalized threshold

% typ type structure

% Version 2.0 - September 2016

function tfstr=bsd\_peakmap\_cell(in,lfft,thr,typ)

% peakmap creation

%

% tfstr=bsd\_peakmap(in,lfft,thr,typ)

%

% in input bsd

% lfft length of the ffts (multiple of 4)

% thr normalized threshold

% typ type structure

% Version 2.0 - September 2016

function peaks=bsd\_peakmap\_conv(tfstrcell)

% converts cell structured peaks to array

%

% Version 2.0 - September 2016

function out=bsd\_tfstr\_db(list,what,par)

% access bsd db for synthetic results on tfstr

%

% out=bsd\_tfstr\_db(list,what,par)

%

% list file list with path

% what items

% 'basic' basic parameters

% 'hdens' t-f h densities

% 'wn' wiener filters

% 'peaks' peak table data

% 'skyp' skypoints parameters

% 'pers' persistence

% 'tfhist' t-f histograms

% par parameter structure

% tim [min max] times

% fr [min max] frequencies

% Version 2.0 - September 2016

function out=bsd\_tfstr\_db\_all(list,filout)

% access bsd db for synthetic results on tfstr

%

% out=bsd\_tfstr\_db(list,what,par)

%

% list file list with path

% filout file to save the structure

% Version 2.0 - September 2016

function out=bsd\_tfstr\_extr(addr,tab,datyp)

% extracts data from tfstr

%

% out=bsd\_tfstr\_extr(addr,tabl,datyp)

%

% addr BSD path(without final dirsep, ex.: 'I:')

% tab BSD sub-table (to create a BSD sub-table use

% [tab\_out,epoch0,maxt]=bsd\_extr\_subtab(tab,tim,fr)

% datyp data type ('hdens' 'wn' 'peaks' 'npeaks' 'persist' 'persist0'

% 'tfhist' 'tfhist0')

BSD collection access

function [bsd\_out,BSD\_tab\_out,out]=bsd\_access(addr,ant,runame,tim,freq,mode,modif,modifpost)

% data access

%

% [bsd\_out,BSD\_tab\_out,out]==bsd\_access(addr,ant,run,tim,freq,mode,modif)

%

% addr path that contains BSD data master directory without the final dirsep

% ant antenna name

% runame run name

% tim [tini tfin] (mjd) ; if length(tim) = 1 the whole run

% freq [frini frfin] ; if length(freq) = 1 the full band with frini

% mode mode = 0 only the output table

% > 0 creates the output bsd

% = 1 automatic

% = 2 sub-band, concatenates files

% = 3 sub-period, many bands

% = 4 inter-bands sub-band

% modif (if present) modification structure: modifies accessed primary files

% e.g. adding signals or sources; see bsd\_acc\_modif

% [] no modification

% modifpost (if present) modification after the band extraction

% Snag Version 2.0 - October 2016

function [bsd\_out,BSD\_str\_out,out]=bsd\_access\_str(addr,ant,run,tim,freq,mode,modif)

% data access (STRUCTURE VERSION for old Matlab)

%

% bsd\_out=bsd\_access\_str(addr,ant,run,tim,freq,mode,modif)

%

% addr path that contains BSD data master directory without the final dirsep

% ant antenna name

% run run name

% tim [tini tfin] (mjd) ; if length(tim) = 1 the whole run

% freq [frini frfin] ; if length(freq) = 1 the full band with frini

% mode mode = 0 only the output table

% > 0 creates the output bsd

% = 1 automatic

% = 2 sub-band, concatenates files

% = 3 sub-period, many bands

% = 4 inter-bands sub-band

% modif (if present) modification structure: modifies accessed primary files

% e.g. adding signals or sources; see bsd\_acc\_modif

% Version 2.0 - November 2016

function out=bsd\_acc\_modif(in,modstr)

% modifies accessed bsd

%

% out=bsd\_acc\_modif(in,modstr)

%

% in input bsd (typically same as out)

% modstr modification structure or a cell array with more than one mod. struct

% .typ 'sinsig' 'source' 'tsig' 'schirp'

% typ 'sinsig' sinusoidal signal

% .t00 .fr .ph .amp

% typ 'source' periodic source

% .sour the source structure, .phs .A

% typ 'tsig' time signal (same in some bands)

% .t00 .shape .bands

% typ 'schirp' super-chirp (fr should grow very slowly)

% .t00 .fr0 .dfr .amp

% typ 'secs' super-chirp (fr should grow very slowly)

% .t00

% Snag Version 2.0 - November 2016

function out=bsd\_large\_band(tab,tin,tfi,addr,outdt)

% reduced bsd\_largerband

%

% out=bsd\_large\_band(tab,tin,tfi,outdt)

%

% tab bsd table

% tin initial time (mjd)

% tfi final time

% addr (used in case of table) path without dirsep; otherwise ''

% outdt (if present) output file sampling time

% Version 2.0 - November 2016

function out=bsd\_interband(addr,tab,band,tim)

% inter-band reconstruction

% Version 2.0 - November 2016

BSD PSS

Here are the procedures for production software. There are some different types of procedures:

* blind search
* direct search
* targeted search

Each procedure can be divided in parallel jobs.

### **Work structures**

There are two info structure that control the procedures and the results: the proc\_info with the general information on the procedure and the job\_info with the info and results on the single job.

proc\_info

.bsdcont

.type ‘blind’ ‘direct’ ‘targeted’ ‘follow-up’

.comp ‘normal’ ‘mex’ ‘parallel’

.epoch

.mode

.hm

.oper ‘adaptive’ ‘noadapt’ ‘noiseadapt’ ‘onlysigadapt’

.fr

.dnat

.enh

.sd

.min

.step

.max

.cand

.ref

.fu

.

job\_info

.proc

.proc\_info

### **Functions**

function [bsd\_pack\_0, bsd\_packs]=BSD\_PREPJOB(list,frmin,runstr,mode)

function [cand,bsd\_summary,check]=BSD\_JOB(bsd\_pack\_0,bsd\_pack,outdir)

function out=bsd\_mode(typ)

% model of mode function

%

% typ type number

% Snag Version 2.0 - January 2017

function [hmap,job\_info,checkE]=bsd\_hough(typ,peaks,proc\_info,job\_info)

% hough map creation

%

% typ(2) 1-> 1 allsky normal, 2 refinement, 3 follow-up or directed

% 2-> 1 normal, 2 mex file, 3 parallel

% peaks(5+,n) peak table (rows with ...)

% proc\_info epoch

% hm.oper ('adapt','noadapt','noiseadapt','onlysigadapt')

% hm.fr (minfr dnat frenh maxfr)

% hm.sd (minsd step nsteps)

% hm.fr (minfr dnat frenh maxfr)

% Adimref=[nsd,nf] if present (refinement)

% job\_info used for output

%

% coreHoughDynLoop\_mex(peaks,inifr,dfr,Day\_inSeconds,deltaf2,ii0,ii,nbin\_d,d,I500,nTimeSteps,nf);

% Snag Version 2.0 - January 2017

function [cand,job\_info,checkF]=bsd\_cand(hmap,proc\_info,job\_info,kcand)

% finds peaks in the hough map

%

% [cand,job\_info,checkF]=bsd\_cand(hmap,proc\_info,job\_info,kcand)

%

% hmap hough map

% proc\_info

% job\_info

% kcand number of primary candidates to be found

%

% cand(9,N) [fr lam bet sd amp CR dlam dbet typ]

% job\_info job info structure

% checkF service structure for test and debug

% Snag Version 2.0 - January 2017

function out=bsd\_fu(in,direct,fupars)

% bsd base follow-up

%

% out=bsd\_fu(in,direct,lenini)

%

% in input bsd (not corrected, possibly small band)

% direct candidate or direct parameters

% .a right ascenson (deg)

% .d declination

% .f frequency and derivative

% fupars follow-up parameters

% .da RA step

% .dd dec step

% .df frequency and derivatives steps

% .na RA number of steps

% .nd dec number of steps

% .nf frequency and derivatives number of steps

% Snag Version 2.0 - January 2017

function out=dopp\_corr\_residual(direc,T,icplot,ant)

% residuals after doppler correction

%

% out=dopp\_corr\_residual(direc,T)

%

% direc direction ([lam bet] or direct structure) or candidate

% T [min max] mjd or bsd

% icplot 1 plot (def 0)

% ant (if T is not a bsd) ant structure

% Snag Version 2.0 - January 2017

## **BSD other searches**

function [tfstr,job\_summary,check]=DIRECT\_JOB(direct,bsdin,job\_direct\_0,job\_direct,outdir)

%

% proc\_info epoch

% fr.dnat

% mode.hm\_job.frenh

% hm.oper ('adapt','noadapt','noiseadapt','onlysigadapt')

% \*\*mimaf0

% hm.sd (minsd step nsteps)

% hm.fr if exists (minfr dnat frenh maxfr)

% Adimref=[nsd,nf] if present (refinement)

% Snag Version 2.0 - January 2017

BSD time events

function [ffrbsd,fil]=bsd\_frfilt(in,frfilt)

% frequency filter for a bsd gd produced by bsd\_largerband

%

% ffrbsd=bsd\_frfilt(in,frfilt)

%

% in input bsd

% frfilt frequency filter or 'white' or 'wiener'

% Version 2.0 - November 2016

function hbsd=bsd\_resenh(in,enh) **% OPER structure**

% bsd resolution enhancement

%

% in input bsd

% enh (>1) resolution enhancement

% Version 2.0 - November 2016

# **BSD noncohe**

The non-coherent analysis methods are collected in the noncohe folder.

## **Typically batch procedures and reanalysis**

### sid\_sweep

function sids=sid\_sweep(addr,ant,runame,freq,direc,wband,sband,icflat,frbase,pcheck)

% analyze data from known direction for sidereal patterns

%

% sids=sid\_sweep(addr,ant,runame,freq,direc,wband,sband,icflat,frbase)

%

% addr,ant,runame as for bsd\_lego

% freq a frequency inside the bsd base band (10 Hz)

% direc direction structure

% wband sub-band division (0.5,1,2,5,10 Hz; typically 1 Hz)

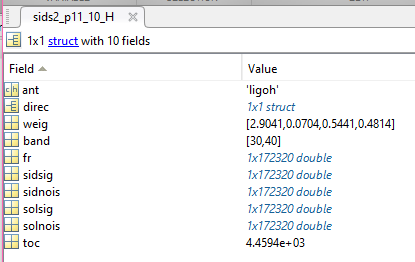
% sband search band (in units of 1/SD; typically 10)

% icflat >0 flat weights (def 0)

% frbase basic bsd frequency (def 10)

% pcheck if present, substitute SD for check (no SOL)

produces



% driver\_sid\_sweep

bw=10; % bw narrow bandwidth in units of 1/SD (typically 10)

freq=10;

for i = 1:70

nameL=sprintf('sidsGC\_%04d\_%04d\_%d\_L',freq,freq+10,bw)

tic

sidsGCL=sid\_sweep('I:','ligol','O2',freq+5,GC,1,10)

save(nameL,'sidsGCL')

toc

nameH=sprintf('sidsGC\_%04d\_%04d\_%d\_H',freq,freq+10,bw)

tic

sidsGCH=sid\_sweep('I:','ligoh','O2',freq+5,GC,1,10)

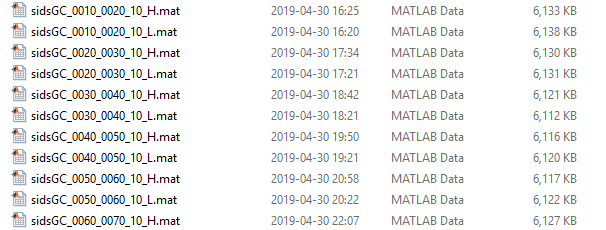
save(nameH,'sidsGCH')

toc

freq=freq+10;

end

Creates files:



% driver\_sid\_sweep1

kpuls=[0 1 2 3 5 6 8 9 10 11];

epochO2=57870;

bw=10; % bw narrow bandwidth in units of 1/SD (typically 10)

for ipul = kpuls

puls=['pulsar\_' num2str(ipul)];

sidsL=['sids2\_p' num2str(ipul) '\_' num2str(bw) '\_L'];

sidsH=['sids2\_p' num2str(ipul) '\_' num2str(bw) '\_H'];

eval([puls '=new\_posfr(' puls ',epochO2);']);

eval(['pfr0=' puls '.f0;'])

eval([sidsL '=sid\_sweep(''I:'',''ligol'',''O2'',' num2str(pfr0) ',' puls ',1,10)'])

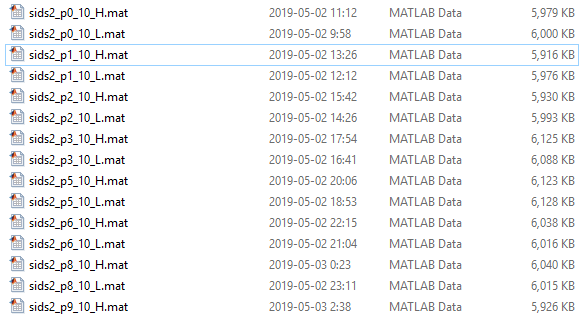
eval(['save(''' sidsL ''',''' sidsL ''');'])

eval([sidsH '=sid\_sweep(''I:'',''ligoh'',''O2'',' num2str(pfr0) ',' puls ',1,10)'])

eval(['save(''' sidsH ''',''' sidsH ''');'])

end

Creates ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­files :



function out=ana\_sid\_sweep1(ipul,bw)

%

% ipul hardware injection number

% bw narrow bandwidth in units of 1/SD (typically 10)

function out=ana\_sid\_sweep2(ipul,bw,integ)

%

% ipul hardware injection number

% bw narrow bandwidth in units of 1/SD (typically 10)

% integ integration values (ex.: [1 3 10 30 100] ; def 1)

function aaout=anana\_sid\_sweep\_GC(thr,range,bw)

% analysis after ana\_sid\_sweep\_GC

%

% thr threshold (def 1.e7)

% range def 10:10:1000

% bw def 10

function aasolout=anana\_sid\_sweep\_GC\_sol(thr,range,bw)

% analysis after ana\_sid\_sweep\_GC on solar data

%

% thr threshold (def 1.e7)

% range def 10:10:1000

% bw def 10

function sids=sid\_sweep\_ref\_wrapper(addr,ant,runame,freq,direc,sband,nsid)

% to apply sid\_sweep\_ref to large bands in 1-Hz subbands

%

% input like sid\_sweep\_ref

% freq any frequency inside the 10 Hz band

% iout is not used

% frbase = 10 (Hz)

% no flat

function done=crea\_sid\_sweep\_db(addr,ant,runame,BAND,direc,sband,nsid)

% creates files collections like

%

% sids\_GC\_O2\_0510\_0520\_L.mat

%

% addr,ant,runame standard inputs

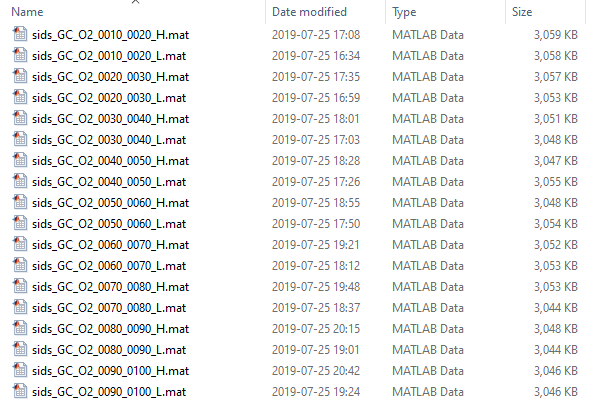
% BAND bands (starting from the BAND(1) base to BAND(2) base)

% direc direction (should have a reasonable name: it will appear in the name of the files)

% sband,nsid see sid\_sweep\_ref\_wrapper

the files are, e.g. with

done=crea\_sid\_sweep\_db('I:','ligol','O2',[530 1020],GC,[10,5])



analized by

function aaout=anana\_sid\_sweep(pref,ants,thr,range)

% analysis after ana\_sid\_sweep

%

% pref prefix (ex.: sids\_GC\_O2\_)

% ants cell array with at least two of 'L' 'H' 'V'

% thr threshold (def 1.e7)

% range def 10:10:1000

## **Typically interactive procedures**

### sid\_sweep\_tf

function sidstf=sid\_sweep\_tf(addr,ant,runame,freq,direc,sband,nSD,outband)

% time-frequency refinement of sid\_sweep candidates

%

% sids=sid\_sweep\_tf(addr,ant,runame,freq,direc,sband)

%

% addr,ant,runame as for bsd\_lego

% if addr is a bsd, use it

% if ant is a table, use it

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% direc direction structure

% sband search band (in units of 1/SD; typically 10)

% if present sband(2) = interlacing delay (in 1/SD)

% nSD time resolution in SD (typically integer); if 2-dim,

% nSD(2) interlacing shift (def nSD(2)=nSD(1)/2)

% nSD(3) resolution enhancement (def 2)

% outband output sub-band

% follow\_cand\_1

ant='ligol';

ant1='L';

pos='\_GC\_';

frcand=[80.03144716

200.2392189

203.9742236

227.7629126

488.3348368

524.3393044

531.1259047

716.0532049

930.3960485

1184.548527

1404.366458

1463.462946

1581.119174

1663.149171

1973.277106

2017.351895

]

frcand(1)=frcand(1)+0.47;

frcand(2)=frcand(2)+0.27;

frcand(9)=frcand(9)+0.17;

ncand=length(frcand);

for i = 13:ncand

candout=[ant1 pos num2str(frcand(i))];

ii=strfind(candout,'.')

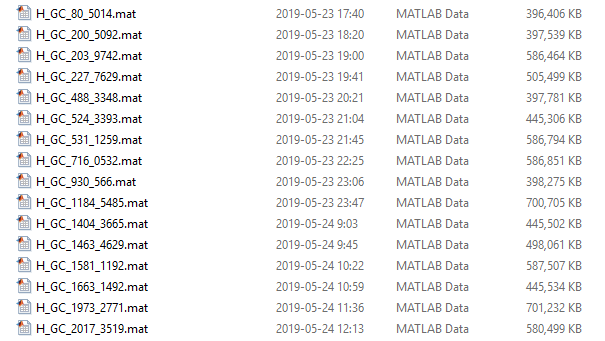
candout(ii)='\_';

eval([candout '=sid\_sweep\_tf(''J:'',''' ant ''',''O2'',frcand(i),GC,[10,5],7);'])

eval(['save(''' candout ''',''' candout ''')'])

end

creates files:



% follow\_hinj\_1

ant='ligol';

ant1='L\_';

epoch=v2mjd([2017 1 1]);

for i = 13:14

puls=['pulsar\_' num2str(i)];

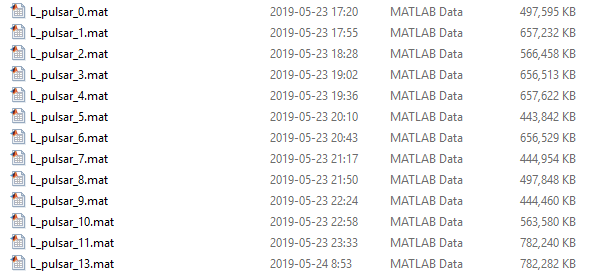
eval([puls 'A=new\_posfr(' puls ',epoch);'])

eval([ant1 puls '=sid\_sweep\_tf(''J:'',''' ant ''',''O2'',' puls 'A.f0,' puls ',[10,5],7);'])

eval(['save(''' ant1 puls ''',''' ant1 puls ''')'])

end

creates files:



### sid\_sweep\_ref

function sidstf=sid\_sweep\_tf(addr,ant,runame,freq,direc,sband,nSD,outband)

% time-frequency refinement of sid\_sweep candidates

%

% sids=sid\_sweep\_tf(addr,ant,runame,freq,direc,sband)

%

% addr,ant,runame as for bsd\_lego

% if addr is a bsd, use it

% if ant is a table, use it

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% direc direction structure

% sband search band (in units of 1/SD; typically 10)

% if present sband(2) = interlacing delay (in 1/SD)

% nSD time resolution in SD (typically integer); if 2-dim,

% nSD(2) interlacing shift (def nSD(2)=nSD(1)/2)

% nSD(3) resolution enhancement (def 2)

% outband output sub-band

### sid\_sweep\_ref\_multisband

function [sidsref,outpars]=sid\_sweep\_ref\_multisband(addr,ant,runame,freq,direc,sbands,interl,check\_sour)

% refinement of sid\_sweep candidates on many sbands

%

% [sidsref,outpars]=sid\_sweep\_ref\_multiband(addr,ant,runame,freq,direc,sbands,interl)

%

% addr,ant,runame as for bsd\_lego

% if addr is a bsd, use it

% if ant is a table, use it

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% direc direction structure

% sbands search bands (in units of 1/SD; typically [10,100,1000])

% interl interlacing delay (in sband units; def 0.5)

% check\_sour if present, for checking

### sid\_sweep\_ref\_multidirec

function [sidsref,outpars]=sid\_sweep\_ref\_multidirec(addr,ant,runame,freq,direcs,sband,check\_sour)

% refinement of sid\_sweep candidates on many sbands

%

% [sidsref,outpars]=sid\_sweep\_ref\_multidirec(addr,ant,runame,freq,direc,sband,check\_sour)

%

% addr,ant,runame as for bsd\_lego

% if addr is a bsd, use it

% if ant is a table, use it

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% direcs cell array with direction structures

% sband search band (in units of 1/SD; typically 10)

% if present sband(2) = interlacing delay (in 1/SD)

% sband(3) = enlargement factor (def enl=10, min recommended 5)

% check\_sour if present, for checking

% driver\_sid\_sweep\_ref\_multidirec

pp=new\_posfr(pulsar\_3,57859);

ppa=pp;

ppb=pp;

ppa.a=pp.a+5;

ppb.d=pp.d+5;

direc{1}=pp;

direc{2}=ppa;

direc{3}=ppb;

[sidsrefdir,outparsdir]=sid\_sweep\_ref\_multidirec('I:','ligol','O2',pp.f0,direc,[10 5],pp)

### sid\_sweep\_multi

function [sids,bsdsstr]=sid\_sweep\_multi(bsdsstr,freq,direc,sband,nsid)

% multi antenna sid\_sweep candidates

%

% sids=sid\_sweep\_multi(bsdsstr,freq,direc,sband,nsid)

%

% bsdsstr bsd search structure array (with addr,ant,runame as for bsd\_lego)

% if it is a cell array containing bsds, use them

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% direc direction structure

% sband search band (in units of 1/SD; typically 10)

% if present sband(2) = interlacing delay (in 1/SD)

% sband(3) = enlargement factor (def enl=10, min recommended 5)

% nsid number of sidereal bins (def 24)

function [anas,selfr]=ana\_multi\_sids(sids,typ)

% analysis of multi\_sids

%

% sids multi\_sid output cell array

% typ type of analysis

% 0 base (def)

### bsd\_noncohe\_full

function [noncohe bsds]=bsd\_noncohe\_full(bsdsstr,band,direc,check\_sour)

% non-coherent full analysis (not survey analysis)

%

% noncohe=bsd\_noncohe\_full(bsdsstr,band,direc,check\_sour)

%

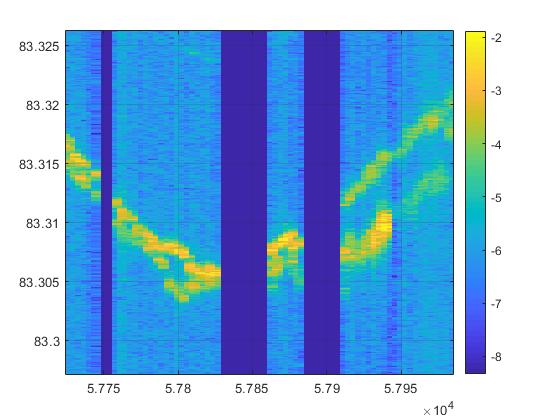
% bsdsstr bsd search structure array (with addr,ant,runame as for bsd\_lego)

% or bsds cell array

% band frequency or band

% direc source direction

% check\_sour inj or known source (if present)



Line at 83 Hz, O2 LL

### Sidereal pattern analysis

function sidpat\_base=ana\_sidpat\_base(ant,direc,N,res)

% sidpat\_base to analyze sidereal pattern

%

% ant antenna (e.g. 'virgo')

% direc direction structure or cw-source structure

% N number of montecarlo samples for background evaluation (def no)

% -1: no background, full range (symmetrical)

% <-1: background with -N, full range (symmetrical)

% res resolution for eta and psi [neta,npsi] (def [201,90])

function out=ana\_sidpat(sidpat,sidpat\_base)

% analyze sidereal pattern

%

% sidpat sidereal pattern (array) or Fourier components abs val (0 1 2 3 4)

% sidpat\_base sidpat\_base, created by ana\_sidpat\_base

function out=ana\_sidpat\_n(ants,sidpats,direc)

% ana\_sidpat for multiple antennas

%

% ants antennas cell array

% sidts sidpats cell array

% direc direction structure

% driver\_ana\_sidpat\_n

pp=new\_posfr(pulsar\_3,57859)

pp=new\_posfr(crab,57859)

pp.eta,pp.psi

sidpat\_baseL=ana\_sidpat\_base(ligol,pp);

sidpat\_baseH=ana\_sidpat\_base(ligoh,pp);

sidpat\_baseV=ana\_sidpat\_base(virgo,pp);

[sidpatL,ftspL,vL]=pss\_sidpat(pp,ligol,240,1);

[sidpatH,ftspH,vH]=pss\_sidpat(pp,ligoh,240,1);

[sidpatV,ftspV,vV]=pss\_sidpat(pp,virgo,240,1);

anaL=ana\_sidpat(y\_gd(sidpatL),sidpat\_baseL);

anaH=ana\_sidpat(y\_gd(sidpatH),sidpat\_baseH);

anaV=ana\_sidpat(y\_gd(sidpatV),sidpat\_baseV);

spb\_fit=anaL.spb\_fit+anaH.spb\_fit+anaV.spb\_fit;

ants{1}=ligol;

ants{2}=ligoh;

ants{3}=virgo;

sidpats{1}=sidpatL;

sidpats{2}=sidpatH;

sidpats{3}=sidpatV;

out=ana\_sidpat\_n(ants,sidpats,pp);

eta1=out.ana{1}.eta0(1);

psi1=out.ana{1}.psi0(1);

eta2=out.ana{2}.eta0(1);

psi2=out.ana{2}.psi0(1);

eta3=out.ana{3}.eta0(1);

psi3=out.ana{3}.psi0(1);

fprintf('%s 0 -> %.2f %.0f 1 -> %.2f %.0f 2 -> %.2f %.0f 3 -> %.2f %.0f \n',pp.name,pp.eta,mod(pp.psi,90),eta1,psi1,eta2,psi2,eta3,psi3)

### Other

% bsd\_block\_noncohe\_frsd

%

% non-coherent procedure to find fr and sd

% needs procedure bsd\_block\_1

% needs data sd1 and sd2

% produces bsd\_out,BSD\_tab\_out,stpar

%

% bsd\_block procedures: copy the script in the work folder

% with different name and modify

% do not modify original block

sd1=-1.e-12;

sd2=0.3e-12;

FS=1/SD;

Dfr=(-2:2)\*FS;

xs=target.f0+Dfr;

[out\_nc,bsd\_ftr]=bsd\_find\_fr\_sd(bsd\_out,target,freq1,[sd1 sd2])

f5v=out\_nc.f5v;

fprintf('Theoretical frequency %.7f Hz found %.7f Hz diff %d \n',xs(3),f5v.frs5v(3),f5v.frs5v(3)-xs(3))

plot\_lines(xs,f5v.fil)

[v,tculm]=bsd\_5vec(bsd\_ftr,xs(3));

hol=bsd\_hole\_window(bsd\_ftr,1);

sid=bsd\_sid(bsd\_ftr,target,100)

[v,tculm]=bsd\_5vec(bsd\_ftr,f5v.frs5v(3));

function [ratmap,intsig]=int\_sid\_sweep(sids,ints,typ,icsol)

% sid\_sweep integration

%

% sids output of a sid\_sweep

% ints integration windows (integer array)

% typ normalization (0 no, 1 yes, 2 sqr (def))

% icsol = 1 solar data (def 0 sidereal data)

### Service

function [map,pars]=fdf\_rh\_map(tfmap,typ,thr,maxslope,epoch)

% fdf hough or radon map

%

% tfmap time-frequency gd2 map

% typ type of output map

% thr threshold (in fraction: 0 all 1 only the maximum)

% maxslope max slope (> 0 ; 0 -> defauult)

% epoch any value, 0 beginning, 1 end

NC procedures

NC is the second version of the BSD noncohe library. The first version will become obsolete. The aspects of the NC are:

* rationalization and simplification
* correction
* velocity
* standardization of output file collection

The library is composed by

* the basic “engine” NC\_sid\_sweep\_base
* first level analysis functions as NC\_sid\_sweep\_multi, NC\_sid\_sweep,…
* batch functions as NC\_crea\_DC1,
* service functions, as NC\_bsdcorr, NC\_filnam,
* data collection analysis, as NC1\_read,

## **Basic engine**

function [outsid,NCcom]=**NC\_sid\_sweep\_base**(bsd\_corr,direc,freq,NCcom)

% basic sid\_sweep

%

% [outsid,NCcom]=NC\_sid\_sweep\_base(bsd\_corr,direc,freq,NCcom)

%

% bsd\_corr corrected bsd (by e.g. NC\_bsdcorr)

% direc direction structure ('nodirec' direction absent)

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% NCcom

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

%

% post postfilter

% == 0 no

% == 1 basic (rectangular length round(sband/5)

% otherwise the MA filter

%

% out.ff 1 all, or indices [ini fin]

% out.POW 1 all, or indices [ini fin]

The output contains:

outsid.CHECKSOLAR=checksolar; can be absent

outsid.start=datetime('now');

outsid.ant=ant;

outsid.t0=t0;

outsid.dt=dt;

outsid.n=n;

outsid.Tobs=Tobs;

outsid.inifr=inifr;

outsid.bandw=bandw;

outsid.dfr=dfr;

outsid.band=band;

outsid.searband=sband(1);

outsid.interl=interl;

outsid.enl=enl;

outsid.i1=i1;

outsid.i2=i2;

outsid.weig=weig;

outsid.wnois=wnois;

outsid.microband=microband;

outsid.DFR=DFR;

outsid.DT=DT;

outsid.maxsidbins=maxsidbins;

outsid.nsid=nsid;

outsid.nweek=nweek;

outsid.dweek=dweek;

if nweek > 0

outsid.Nweek=Nweek;

end

outsid.NW=NW;

outsid.dNW=dNW;

outsid.post=post;

outsid.frdel=frdel;

outsid.fr=fr-frdel;

outsid.k1=k1;

outsid.K1=K1;

outsid.k2=k2;

outsid.N=N;

outsid.Y1=mean(abs(Y').^2);

outsid.Y2=mean(abs(Y).^2);

outsid.cover=cover;

for non-week mode:

outsid.sidsig=sidsig;

outsid.sidnois=sidnois;

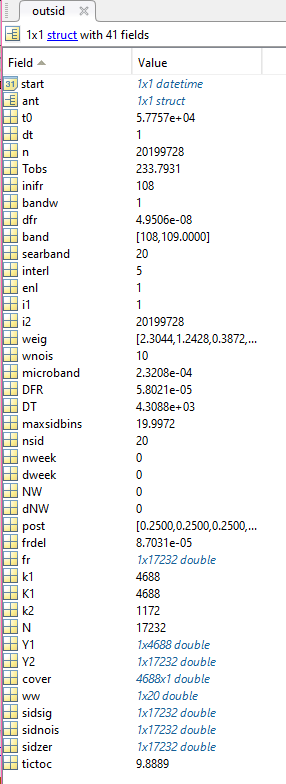
outsid.sidzer=sidzer;

for week mode:

outsid.sidsigs=sidsigs;

outsid.sidnoiss=sidnoiss;

outsid.sidzers=sidzers;



This function produces 3 arrays with the “signal”, the “sidereal noise” and the “non-sidereal noise”.

It can be used in two modes: noweek (only frequency) and week (time-frequency).

Solar periodicities can be checked.

## First level analysis

function [sids,bsdsstr]=NC\_sid\_sweep\_multi(bsdsstr,direc,freq,NCcom)

% multi antenna sid\_sweep candidates

%

% sids=sid\_sweep\_multi(bsdsstr,direc,freq,NCcom)

%

% bsdsstr bsd search structure array (with addr,ant,runame as for bsd\_lego)

% if it is a cell array containing bsds, use them

% direc direction structure

% freq candidate frequency;

% if 2-dim freq(2), spin-down

% if 3-dim freq(3) is the semiband; def semiband 0.5 Hz

% NCcom

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

%

% out.ff 1 all, or indices [ini fin]

% out.POW 1 all, or indices [ini fin]

% use\_multi

p3=new\_posfr(pulsar\_3,57859);

bsdsstr(1).addr='D:';

bsdsstr(1).ant='ligol';

bsdsstr(1).runame='O2';

bsdsstr(2).addr='D:';

bsdsstr(2).ant='ligoh';

bsdsstr(2).runame='O2';

[sids,bsds]=NC\_sid\_sweep\_multi(bsdsstr,p3.f0,p3,[20 5 1])

% [sids,bsds]=NC\_sid\_sweep\_multi(bsdsstr,[108.5,0,0.5],p3,[20 5 1])

function sidsw=NC\_sid\_sweep(addr,ant,runame,direc,freq,NCcom)

% sid sweep on a band

%

% sidsw=NC\_sid\_sweep(addr,ant,runame,direc,freq,NCcom)

%

% addr,ant,runame as for bsd\_lego

% direc direction structure ('nodirec' direction absent)

% freq a frequency inside the bsd base band (10 Hz)

%

% NCcom

% frbase bsd frequency base (def 10)

% wband sub-band division (0.5,1,2,5,10 Hz; default 1 Hz)

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

## Batch functions

function out=NC\_crea\_DC1(addr,ant,runame,direc,band,header,NCcom)

% creates a type 1 non-coherent DB

%

% out=NC\_crea\_DC1(addr,ant,runame,direc,band,header,NCcom)

%

% addr,ant,runame as for bsd\_lego

% direc direction structure

% band min,max frequency bands (any value inside the first and the second

% header of the files (ex: 'GC')

%

% NCcom

% frbase bsd frequency base (def 10)

% wband sub-band division (0.5,1,2,5,10 Hz; default 1 Hz)

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

function out=NC\_crea\_DC2(addr,ant,runame,direc,band,header,NCcom)

% creates a type 1 non-coherent DB

%

% out=NC\_crea\_DC2(addr,ant,runame,direc,band,header,NCcom)

%

% addr,ant,runame as for bsd\_lego

% direc direction structure

% band min,max frequency bands (any value inside the first and the second

% header of the files (ex: 'GC')

%

% NCcom

% frbase bsd frequency base (def 10)

% wband sub-band division (0.5,1,2,5,10 Hz; default 1 Hz)

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

function outmul=NC\_crea\_DC3(bsdsstr,cands,header,tail,NCcom)

% creates a type 3 non-coherent DB

%

% outmul=NC\_crea\_DC3(bsdsstr,cands,header,tail,NCcom)

%

% bsdsstr bsd search structure array (with addr,ant,runame as for bsd\_lego)

% if it is a cell array containing bsds, use them

% cands list of sources (cell array)

% header of the files (ex: 'GC')

% tail filnam tail

%

% NCcom

% frbase bsd frequency base (def 10)

% wband sub-band division (0.5,1,2,5,10 Hz; default 1 Hz)

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

function outmul=NC\_crea\_DC4(bsdsstr,cands,header,tail,NCcom)

% creates a type 4 non-coherent DB

%

% outmul=NC\_crea\_DC4(bsdsstr,cands,header,tail,NCcom)

%

% bsdsstr bsd search structure array (with addr,ant,runame as for bsd\_lego)

% if it is a cell array containing bsds, use them

% cands list of sources (cell array)

% header of the files (ex: 'GC')

% tail filnam tail

%

% NCcom

% frbase bsd frequency base (def 10)

% wband sub-band division (0.5,1,2,5,10 Hz; default 1 Hz)

% sband search band (in units of 1/SD; default 20) sband=0 -> 20

% interl interlacing delay (in 1/SD); default sband/2, if sband = 20, interl = 5

% enl enlargement factor (def 1)

% nsid number of sidereal bins (def or 0 = round(SD/outsid.DT))

%

% sol if exists and >0 -> also solar (NYI)

% checksolar if exist and = 1, substitutes SD with SolD

% if > 1, substitutes SD with checksolar

%

% nweek if present and >0, t-fr mode, n week for pieces (typically 1)

% dweek week interlacing delay (<= nweek, def nweek/2, =nweek or 0 no interlacing)

## Service functions

function bsd\_corr=NC\_bsdcorr(addr,ant,runame,band,direc)

% creates Doppler corrected bsds

%

% bsd\_corr=NC\_bsdcorr(addr,ant,runame,direc)

%

% addr,ant,runame as for bsd\_lego

% band [frini frfin] ; if length(freq) = 1 or frini=frfin

% the full band with frini (def ortho-band)

% direc direction structure ('nodirec' or 0 no correction)

function filnam=NC\_filnam(typ,header,runame,ant,band,dats,tail)

% name of NC collections files

%

% typ 1 DB1

% header ex.: 'DB1\_GC'

% ant ex.: 'virgo'

% band [min max]

% dats other (1 or more numbers)

% tail suffix (if present)

## Data collection analysis

function [par,sig,nois,zer]=NC1\_read(addr,ant,runame,head,band,plt)

% reads data from NC1 file collection

%

% [out,par]=NC1\_read(addr,ant,runame,head,band)

%

% addr,ant,runame as for bsd\_lego

% head file name header (ex.: 'GC')

% band min,max frequency

% plt > 0, plots

function [par,sigs,noiss,zers,sig,nois,zer]=NC2\_read(addr,ant,runame,head,band,plt)

% reads data from NC1 file collection

%

% [out,par]=NC1\_read(addr,ant,runame,head,band)

%

% addr,ant,runame as for bsd\_lego

% head file name header (ex.: 'GC')

% band min,max frequency

% plt > 0, plots

function out=NC4\_read(fil,kcand,typ,plt)

% reads data from NC3 or NC4 file

%

% out=NC4\_read(fil,kcand,typ,plt)

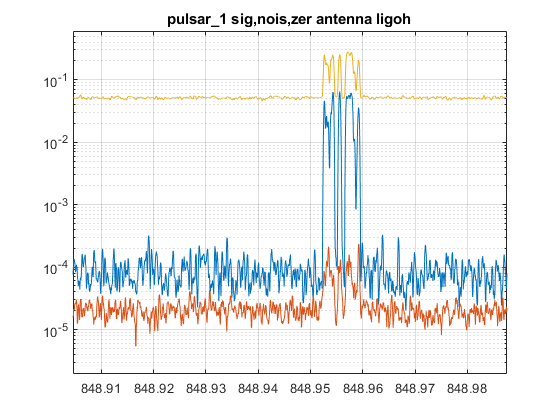
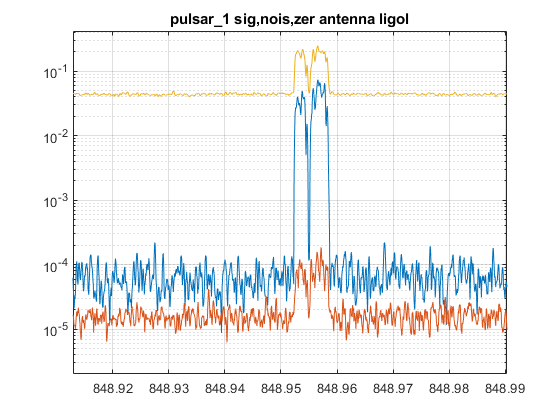
%

% fil type NC3 or NC4 file

% kcand candidate number (0 or absent, list)

% typ 1 only frequency, 2 time frequency

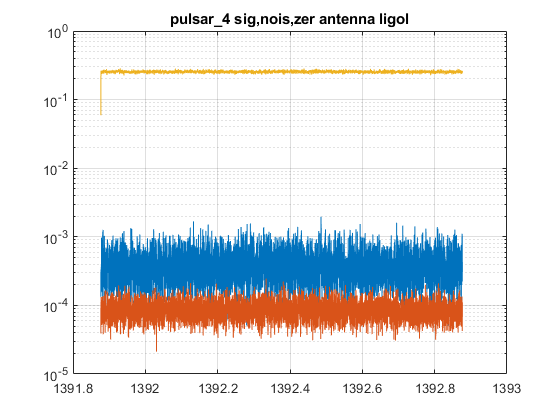
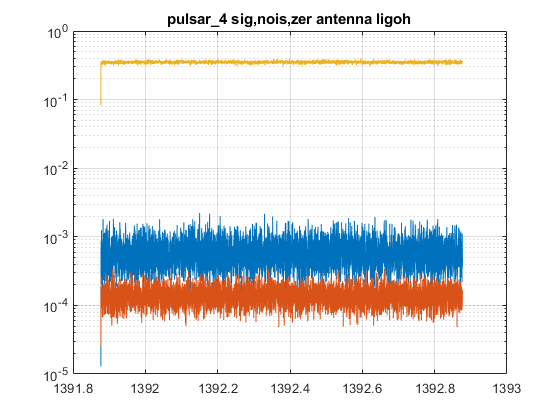
% plt plot type (def no)

Immagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

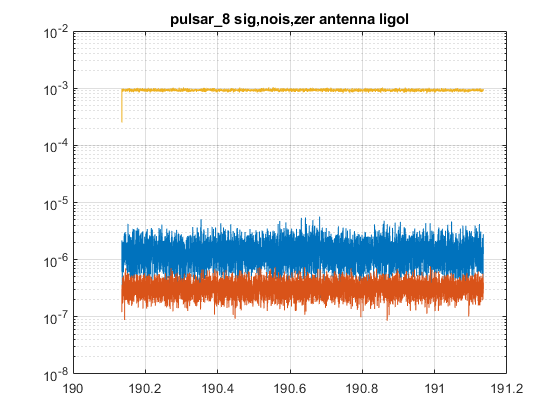
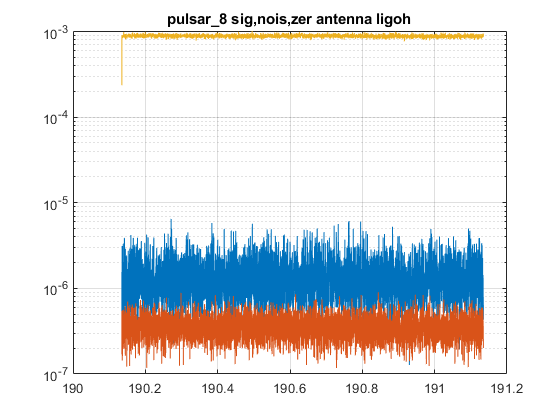
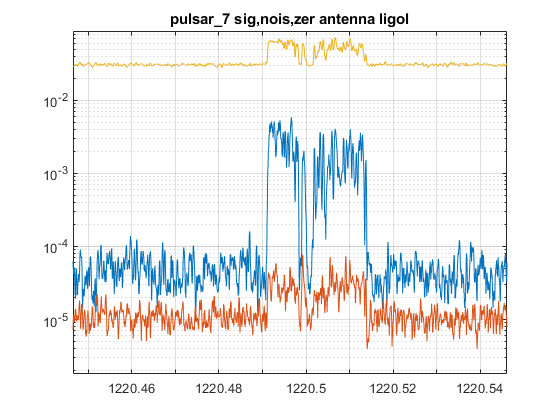
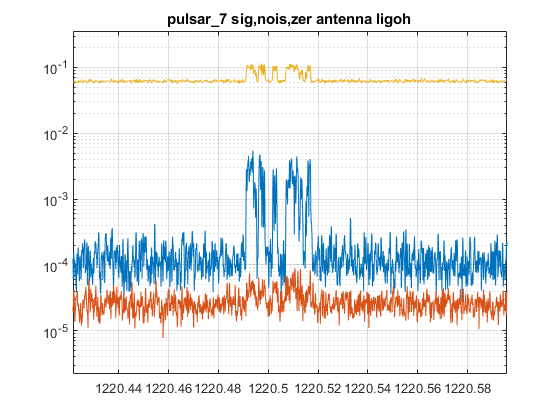
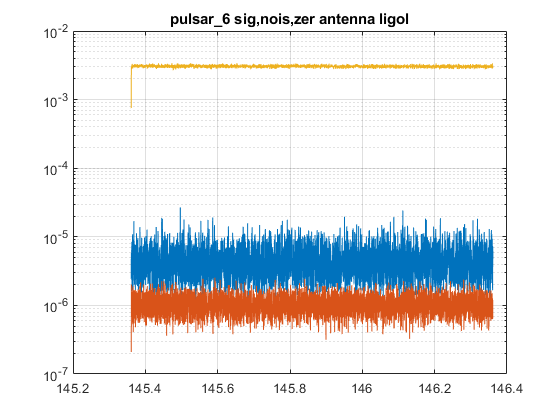
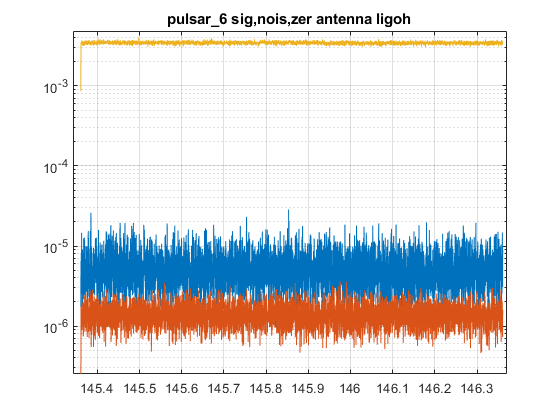
Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

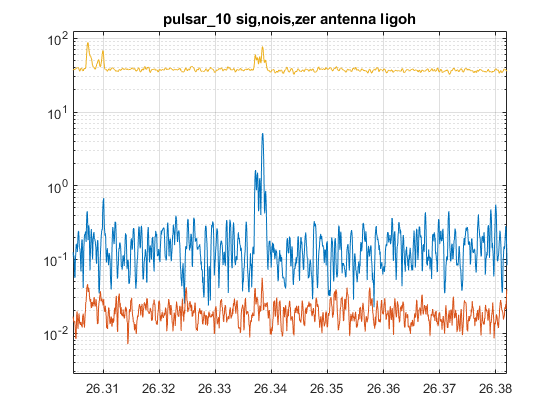
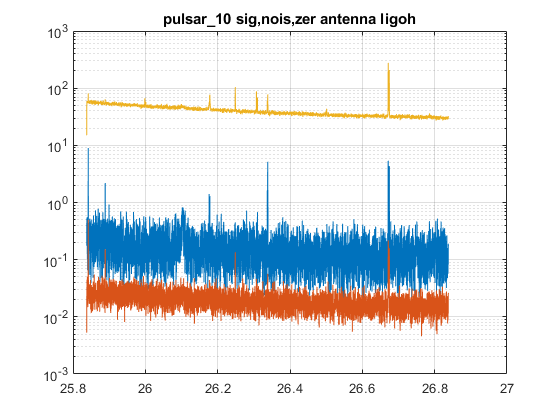
Descrizione generata automaticamenteImmagine che contiene testo, mappa

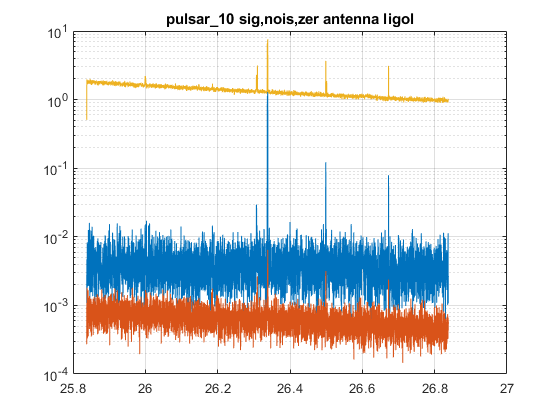
Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

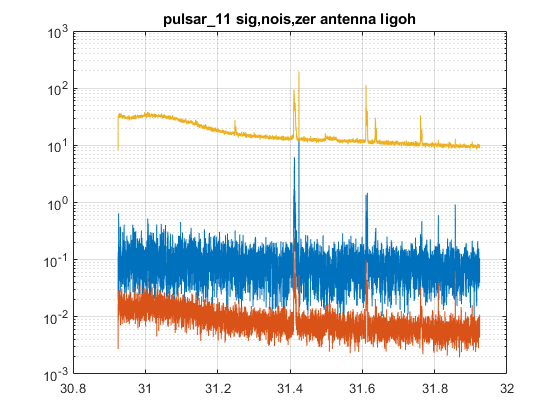
Descrizione generata automaticamenteImmagine che contiene testo, mappa

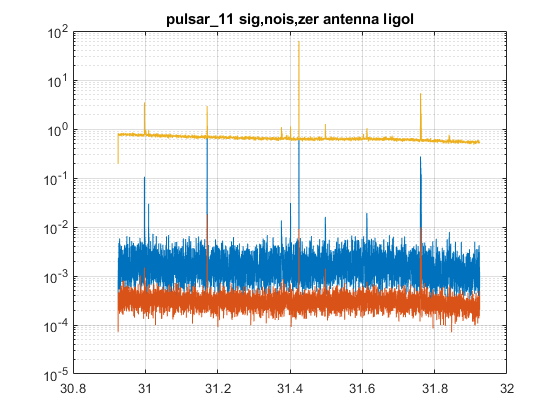
Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene testo, mappa

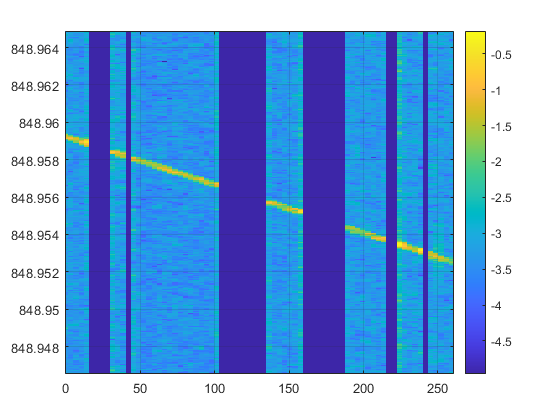
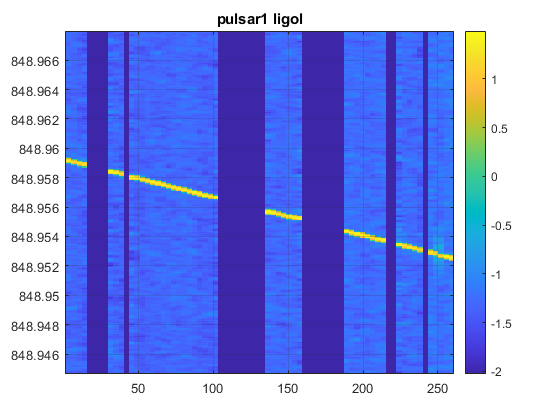
Descrizione generata automaticamenteImmagine che contiene mappa, testo

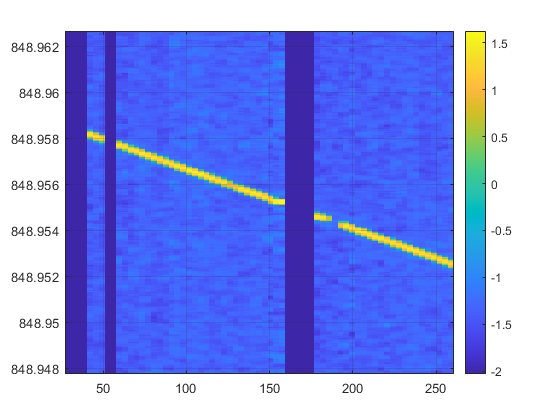
Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamenteImmagine che contiene mappa, testo

Descrizione generata automaticamenteImmagine che contiene testo, mappa

Descrizione generata automaticamente

Immagine che contiene computer

Descrizione generata automaticamenteImmagine che contiene orologio

Descrizione generata automaticamenteImmagine che contiene orologio

Descrizione generata automaticamente

BSD GUIs

function bsds\_addr\_tab=bsds\_gui()

% gui to create bsds\_addr\_tab

% Snag Version 2.0 - April 2021

% show\_multiant

prompt={'dt0'...

'equifreq'...

'orthoband'...

'no T\_mean'...

'no correction'...

'spectral resolution (or 0)'...

'raw out'...

'frequency shift'...

'Output bandwidth'...

'Output structure name'};

defarg={'0.1' '1' '1' '0' '0' '2' '0' '0' '0.5' 'MAout'};

answ=inputdlg(prompt,'Multi Antenna parameters',[1,40],defarg);

MAcom.dt0=eval(answ{1});

MAcom.equifreq=eval(answ{2});

MAcom.orthoband=eval(answ{3});

MAcom.noT\_mean=eval(answ{4});

MAcom.nocorr=eval(answ{5});

MAcom.specres=eval(answ{6});

MAcom.rawout=eval(answ{7});

MAcom.frshift=eval(answ{8});

wband=eval(answ{9});

outstnam=answ{10};

eval([outstnam '=bsd\_multiant(bsds\_addr\_tab,sour,wband,MAcom)'])

BSD procedures

% addobba3

anabasic.tfh\_thr=95;

anabasic.noplot=1;

% list='list\_O1\_L\_raw.txt';

%

% outL=bsd\_tfstr\_add(list,anabasic)

list='list\_O1\_L\_sccl.txt';

outH=bsd\_tfstr\_add(list,anabasic)

% addobba\_notch

anabasic.tfh\_thr=95;

anabasic.noplot=1;

lines=ligo\_readlines('O2L1lines.csv');

parnotch.lines=lines;

parnotch.thr=0.25;

parnotch.win=1;

% list='list\_O1\_L\_raw.txt';

%

% outL=bsd\_tfstr\_add(list,anabasic)

list='list\_O2\_L\_sccl.txt';

outH=bsd\_tfstr\_addwithnotch(list,parnotch,anabasic)

% add\_notch\_to\_tfstr\_files

%

% work in a different folder than the source files

lines=ligo\_readlines('O2L1lines.csv');

parnotch.lines=lines;

parnotch.thr=0.25;

parnotch.win=1;

list='list\_O2\_L\_tfstr.txt';

fidlist=fopen(list,'r');

nfiles=0;

nofiles=0;

while (feof(fidlist) ~= 1)

str=fgetl(fidlist);

str1=str(1);

if strcmp(str1,'\*')

nofiles=nofiles+1;

fprintf(' %s NOT CONSIDERED \n',str);

continue

else

nfiles=nfiles+1;

file{nfiles}=str;

fprintf(' %s \n',str);

end

end

nfiles,nofiles

for i = 1:nfiles

[pathstr,name,ext]=fileparts(file{i});

load(file{i});

name

eval([name '=bsd\_notch(' name ',parnotch);'])

eval(['save(''' name ''',''' name ''',''-v7.3'')'])

eval(['clear ' name])

end

% sHapplier

% gdin=g2

gdin=lvela;

ant=ligol;

name='outLvela'

sour=vela;

cont=cont\_gd(gdin);

t0=cont.t0;

sournew=new\_posfr(sour,t0)

fr=sournew.f0;

fr=[sournew.f0,sournew.df0,sournew.ddf0];

VPstr=extr\_velpos\_gd(gdin)

[pO,vO,tO,psO,vsO,ttO]=interp\_VP(VPstr,sour);

eval([name '\_5vec=sa\_spec5vec(sournew,ant,-1);'])

eval(['[' name ',corr]=vfs\_subhet\_pos(gdin,fr,pO);'])

eval(['s' name '=gd\_pows(' name ');'])

eval(['figure;semilogy(' 's' name '); grid on'])

N=n\_gd(gdin);

dt=dx\_gd(gdin);

gdpar=[dt,N];

sdpar=fr(2:length(fr));

eval('sd=vfs\_spindown(gdpar,sdpar,1);')

eval([name '\_sd=vfs\_subhet(' name ',sd)'])

eval(['s' name '\_sd=gd\_pows(' name '\_sd);'])

fs=sa\_spec5vec(sournew,ant,-1);

eval(['figure;semilogy(' 's' name '\_sd); grid on'])

eval(['plot\_lines(fs(1,:),' 's' name '\_sd);'])

# BSD analysis procedures

function s=bsd\_pows(bsd,res,pieces,windn)

% bsd power spectrum

%

% bsd input bsd

% res resolution

% pieces number of pieces (def 1)

% windn window number (0 -> no, 1 -> bartlett, 2 -> hanning, 3 -> flatcos, 4 -> tukey)

% def hanning

% Snag Version 2.0 - December 2016

function w=bsd\_worm(bsd,freq,icshow)

% worm for bsd

%

% bsd bsd input

% freq frequency (not aliased)

% icshow = 1 plot

% Snag Version 2.0 - December 2016

function v=bsd\_5vec(bsd,frs)

% computes the 5-vec for a bsd

%

% out=bsd\_5vec(bsd,frs)

%

% bsd input bsd

% frs frequencies

% Snag Version 2.0 - December 2016

function out=bsd\_piece\_5vec(in,fr,dt1,dt2,rasc)

% computes the 5-vect for pieces

%

% out=bsd\_piece\_5vec(in,fr,dt1,dt2)

%

% in input bsd

% fr frequency

% dt1 distance between pieces (in SidDays)

% dt2 length of pieces (in SidDays)

% Snag version 2.0 - January 2017

function out=bsd\_specfilt(in,sourd,res)

% applies spectral filter to a bsd

%

% out=bsd\_specfilt(in,sour,res)

%

% in input bsd

% sourd source declination (deg) or source

% res resolution

% Snag Version 2.0 - January 2017

function out=bsd\_sid(in,fr,nsid)

% sidereal analysis (gmst)

%

% in input bsd

% fr frequency

% nsid length of the

% Snag Version 2.0 - January 2017

function [s,pieces,lenpiec]=bsd\_lowres\_pows(bsd,res,lcohe,windn)

% low resolution spectrum

%

% s=bsd\_lowres\_pows(bsd,res,lcohe,windn)

%

% bsd input bsd

% res 0-padding resolution enhancement

% lcohe coherence length (days or s)

% windn window number (0 -> no, 1 -> bartlett, 2 -> hanning, 3 -> flatcos, 4 -> tukey)

% def hanning

% Snag Version 2.0 - January 2017

function out=check\_inj(in,sour,ichole)

% Check injection

%

% in input bsd (not corrected)

% sour source or injection

% ichole > 0 insert input holes (def 1)

% Snag Version 2.0 - February 2017

# Non-bsd useful functions

## Service

function out=stat\_nonzero(gin)

% statistics on non-zero data

%

% out=stat\_nonzero(gin)

%

% gin arragin, gd or gd2

% Snag Version 2.0 - November 2016

function tfstr=tfstr\_gd(gin)

% extracts tfstr structure

% Version 2.0 - September 2016

function [y,mi]=logarithmize(x)

% reduce drastically the dynamics

%

% [y,mi]=logarithmize(x)

%

% x input gd or array

%

% mi median abs level (\*10 typical threshold)

% Version 2.0 - April 2016

function cl=complex\_clip(in,thr)

% puts to zero values of a complex gd with higher than thr abs values

%

% cl=complex\_clip(in,thr)

%

% Snag Version 2.0 - April 2016

function out=extr\_band(in,band,st)

% extract band (output signal is complex)

% out=extr\_band(in,band,st)

%

% in input gd (type 1)

% band [min max] frequency

% st sampling time (if present; otherwise st=1/band)

% Snag Version 2.0 - June 2016

function out=double\_threshold(in,th1,th2,amp)

% double threshold function

%

% out=double\_threshold(in,th1,th2,amp)

%

% in input array or gd or gd2

% th1 exclusion threshold

% th2 saturation threshold

% amp max amplitude (def 1)

% Version 2.0 - February 2017

## VFS (Varying Frequency Signal)

* **vfs\_create**

function [gout,fr,lfr,fr0,ph0]=vfs\_create(n,dt,pols,sins,inp)

% create varying frequency signals

%

% [gout,fr,lfr,fr0,ph0]=vfs\_create(n,dt,pols,sins,inp)

%

% n number of samples

% dt sampling time (negative -> complex signal)

% pols polinomial coefficient (1 fr0, 2 sd, ...)

% pols(1) can be complex (fr0+j\*ph0 (ph0 in deg))

% sins sinusoids (m,3) (A,fr,ph(deg))

% inp irregular input frequency (if exist; will be resampled to n)

%

% ex.: [gout,fr,lfr,fr0,ph0]=vfs\_create(10000,-0.1,1,[0.1,0.01,0]);

function gout=vfs\_create\_1(n,dt,fr0,frph,ic)

% create varying frequency signals

%

% gout=vfs\_create\_1(n,dt,fr0,frph,ic)

%

% n number of samples

% dt sampling time (negative -> complex signal)

% fr0 main frequency

% frph varying frequency or phase

% ic = 0 frequecy additive

% = 1 frequency multiplicative

% = 2 phase

function gout=vfs\_create\_2(gin,frph,ic)

% create varying frequency signals

%

% gout=vfs\_create\_1(n,dt,fr0,frph,ic)

%

% gin input gd analytic signal

% frph varying frequency or phase

% ic = 0 frequecy

% = 1 phase

* **vfs\_freq**

function [fr,fr1,fr2,dy,dy1,dy2]=vfs\_freq(in,cal)

% frequency evaluation

%

% in input gd (complex (if real, it is converted to analytical signal)

% cal > 0 -> calibrated (only on fr), > 1 calibration curve

* **vfs\_subhet**

function [out,lfr1]=vfs\_subhet(in,lfr)

% application of sub-heterodyne

%

% [out,lfr]=vfs\_subhet(in,lfr)

%

% in input data gd

% lfr sub-het varying frequency

function out=vfs\_subhet\_pos(in,fr,pos)

% application of sub-heterodyne

%

% [out,lfr]=vfs\_subhet\_pos(in,fr,pos)

%

% in input data gd

% fr frequency (fixed or varying as fr(1)\*(1+dfr\*t))

% pos position (normalized by c)

* **vfs\_spindown**

function sd= (gpar,sdpar,ic)

% creation/correction spin-down frequency variation

%

% sd=vfs\_spindown(gin,sdpar,ic)

%

% gpar data gd dt and n

% sdpar spin-down parameters array at epoch ini of the gd

% ic 1/-1 creation/correction

function spfilt=crea\_spec\_filter(sour,ant)

% computes the mean spectral filter

%

% out=spec\_filter(sour,ant)

% Version 2.0 - July 2016

## MC Filter

function out=mc\_5vec(ant,sour,N,randph,icplot)

% montecarlo with 5-vect

%

% ant antenna (0 -> all parameters

% ant struct -> fixed antenna)

% sour source (0 -> all parameters

% [alpha,delta] -> fixed position

% sour struct -> fixed source)

% N MC dimension

% randph random phase (source RA: 0 or 1; def 0)

function MCF=crea\_mc\_filter(ant,sour,N,randph)

% create a

%

% ant antenna (0 -> all parameters

% ant struct -> fixed antenna)

% sour source (0 -> all parameters

% [alpha,delta] -> fixed position

% sour struct -> fixed source)

% N MC dimension

% randph random phase (source RA: 0 or 1; def 0)

function out=mc\_filter(MCF,in,ic)

% 5-vect montecarlo filtering

%

% MCF mc filter bank [5,N]

% in input 5-vects [5,n]

% ic plot level

function [sig,nois,sigeta,sigpsi]=mc\_sn\_5vec(ant,sour,N,randph)

% montecarlo with 5-vect

%

% ant antenna (0 -> all parameters

% ant struct -> fixed antenna)

% sour source (0 -> all parameters

% [alpha,delta] -> fixed position

% sour struct -> fixed source)

% N MC dimension

% randph random phase (source RA: 0 or 1; def 0)

% test\_mcf\_10

%

% spectral filter and mc filter

% testing also flat filter and max filter

# Example scripts

## Extracting events

% ex\_bsd\_event

%

% data extraction in a band [30 300] Hz

% ±5 min around the exact minute

%

% whitening filter and time resolution enhancement

% complex to real conversion

tic

MJD=57279.410247905;

mjd\_day=floor(MJD);

mjd\_min=floor((MJD-mjd\_day)\*1440)/1440;

min5=5/1440;

t1=mjd\_day+mjd\_min-min5; mjd2s(t1)

t2=mjd\_day+mjd\_min+min5; mjd2s(t2)

[bsd\_outL,BSD\_tab\_outL,outL]=bsd\_access('I:','ligol','O1',[t1 t2],[30 300],3);

[bsd\_outH,BSD\_tab\_outH,outH]=bsd\_access('I:','ligoh','O1',[t1 t2],[30 300],3);

disp('Data extraction'),toc

[whL,filthL]=bsd\_frfilt(bsd\_outL,'white');

[whH,filthH]=bsd\_frfilt(bsd\_outH,'white');

figure;plot(whL),hold on,plot(whH,'r');

WHL=bsd\_resenh(whL,10);

WHH=bsd\_resenh(whH,10);

figure;plot(WHL);hold on;plot(WHH,'r');plot(whL,'r.');plot(whH,'b.')

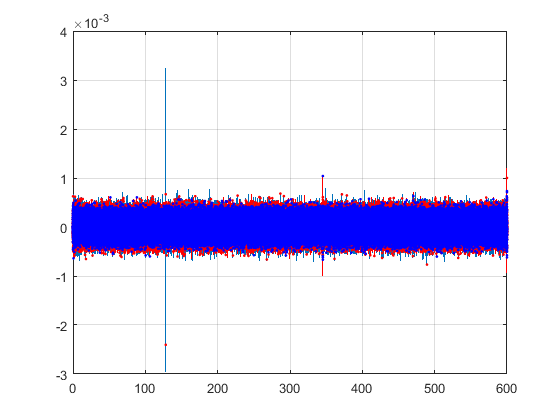
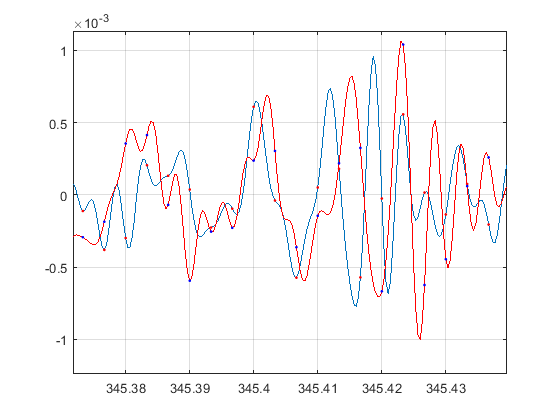
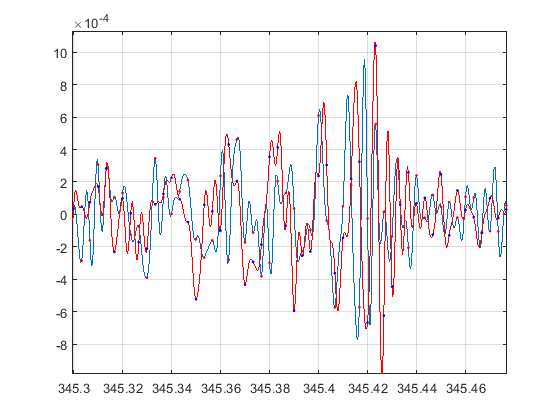
rWHL=bsd\_real(WHL);

rWHH=bsd\_real(WHH);

figure;plot(rWHL);hold on;plot(rWHH,'r');

mjd2s(t1),mjd2s(t2)

disp('End'),toc

****

## Extracting narrow bands

% ex\_bsd\_sband

%

% extracts a band around the pulsar\_3

tic

pp=pulsar\_3;

fr=pp.f0;

dfr=0.1;

[bsd\_outL,BSD\_tab\_outL,outL]=bsd\_access('I:','ligol','O1',[57270 57570],[fr-dfr fr+dfr],2);

[bsd\_outH,BSD\_tab\_outH,outH]=bsd\_access('I:','ligoh','O1',[57270 57570],[fr-dfr fr+dfr],2);

toc

## Correction of Doppler and spin-down

% ex\_bsd\_dedop

%

% after ex\_bsd\_sband

tic

pp0=pulsar\_3;

SD=86164.09053083288;

cont=cont\_gd(bsd\_outL);

t0=cont.t0;

fr0=cont.inifr;

% x=x\_gd(bsd\_outL);

pp=new\_posfr(pp0,t0);

Dfr=pp.f0-pp0.f0;

bsd\_corrL=bsd\_dopp\_sd(bsd\_outL,pp);

bsd\_corrH=bsd\_dopp\_sd(bsd\_outH,pp);

sL=gd\_pows(bsd\_corrL,'resolution',6);

sH=gd\_pows(bsd\_corrH,'resolution',6);

figure,plot(sL,sH)

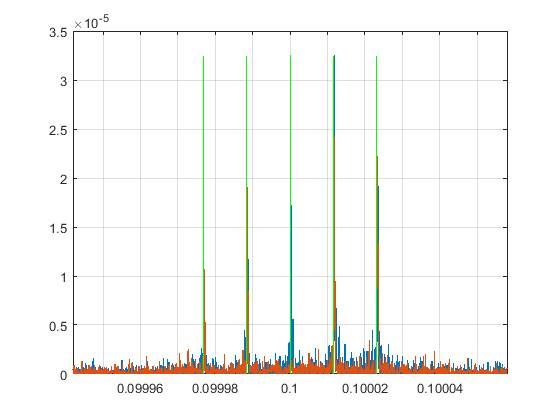
frcent=pp.f0-fr0;

xs=frcent+[-2 -1 0 1 2]/SD;

figure,plot(sL,sH),plot\_lines(xs,sL,'g')

xlim([frcent-5/SD frcent+5/SD])

toc

****

% ex\_bsd\_dedop\_sing

%

% from the bsd primary file

tic

pp0=pulsar\_3;

SD=86164.09053083288;

cont=cont\_gd(L\_C01\_20151112\_0100\_0110\_tfstr);

t0=cont.t0;

fr0=cont.inifr;

pp=new\_posfr(pp0,t0);

Dfr=pp.f0-pp0.f0;

bsd\_corrL=bsd\_dopp\_sd(L\_C01\_20151112\_0100\_0110\_tfstr,pp);

sL=gd\_pows(bsd\_corrL,'resolution',2);

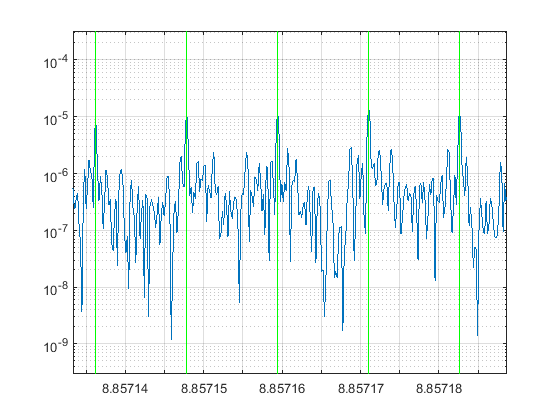
frcent=pp.f0-fr0;

xs=frcent+[-2 -1 0 1 2]/SD;

figure,semilogy(sL),plot\_lines(xs,sL,'g')

xlim([frcent-5/SD frcent+5/SD]),grid on

toc

****

## Software injections

Using bsd\_acc\_modif, internally to bsd\_access and load\_tab\_bsd.

% ex\_bsd\_modif

tic

dfr=0.1;

modif.typ='sinsig';

modif.t00=57270;

modif.fr=108.8;

modif.ph=0;

modif.amp=1;

% modif=[];

modifpost.typ='sinsig';

modifpost.t00=57270;

modifpost.fr=108.8;

modifpost.ph=0;

modifpost.amp=1;

modifpost=[];

fr=modif.fr;

[bsd\_outL,BSD\_tab\_outL,outL]=bsd\_access('I:','ligol','O1',[57270 57570],[fr-dfr fr+dfr],2,modif,modifpost);

toc

% ex\_bsd\_softinj

tic

SD=86164.09053083288;

dfr=0.1;

modif.typ='source';

pp=pulsar\_3;

pp.f0=pp.f0-1;

modif.sour=pp;

modif.phs=[0 0 0];

modif.A=1;

% modif=[];

modifpost.typ='source';

modifpost.sour=pp;

modifpost.phs=[0 0 0];

modifpost.A=1;

modifpost=[];

fr=pp.f0;

[bsd\_sinj\_L,BSD\_tab\_outL,outL]=bsd\_access('I:','ligol','O1',[57270 57570],[fr-dfr fr+dfr],2,modif,modifpost);

bsd\_corrsinj\_L=bsd\_dopp\_sd(bsd\_sinj\_L,pp);

cont=cont\_gd(bsd\_sinj\_L);

t0=cont.t0;

fr0=cont.inifr;

sL=gd\_pows(bsd\_corrsinj\_L,'resolution',6);

figure,plot(sL)

frcent=pp.f0-fr0;

xs=frcent+[-2 -1 0 1 2]/SD;

figure,plot(sL),plot\_lines(xs,sL,'g')

xlim([frcent-5/SD frcent+5/SD])

toc

## Compute 5-vec

% ex\_bsd\_5vec

%

% after ex\_bsd\_sband and ex\_bsd\_dedop

bsd=bsd\_corrL;

tic

t0\_O1=57277;

pp=pulsar\_3; % not updated sources !

pp=new\_posfr(pp,t0\_O1);

f0=pp.f0;

df=1.e-8;

frs=(-15:15)\*df+f0;

n=length(frs);

v=bsd\_5vec(bsd,frs);

for i = 1:n

vnorm(i)=norm(v(:,i));

end

vabs=abs(v);

vang=angle(v)\*180/pi;

for j = 1:5

vang1(j,:)=atan3(v(j,:));

end

frs1=frs(1:n-1)+df/2;

figure,plot(frs,vnorm),grid on,title('Norm')

plot\_lines(f0,vnorm,'g')

[vmax,imax]=max(vnorm);

fmax=frs(imax);

plot\_lines(fmax,vnorm,'r')

figure,plot(frs,vabs(1,:),'r'),grid on

hold on,plot(frs,vabs(2,:),'m')

hold on,plot(frs,vabs(3,:),'k')

hold on,plot(frs,vabs(4,:),'g')

hold on,plot(frs,vabs(5,:),'b'),title('Absolute values')

vabsmax=max(vabs);

plot\_lines(f0,vabsmax,'g')

plot\_lines(fmax,vabsmax,'r')

figure,plot(frs,vabs(1,:)./vabsmax,'r'),grid on

hold on,plot(frs,vabs(2,:)./vabsmax,'m')

hold on,plot(frs,vabs(3,:)./vabsmax,'k')

hold on,plot(frs,vabs(4,:)./vabsmax,'g')

hold on,plot(frs,vabs(5,:)./vabsmax,'b'),title('Normalised absolute values')

vabsmax=max(vabs);

plot\_lines(f0,[0 1.05],'g')

plot\_lines(fmax,[0 1.05],'r')

figure,plot(frs,vang(1,:),'r'),grid on

hold on,plot(frs,vang(2,:),'m')

hold on,plot(frs,vang(3,:),'k')

hold on,plot(frs,vang(4,:),'g')

hold on,plot(frs,vang(5,:),'b'),title('Angles')

plot\_lines(f0,vang(:),'g')

plot\_lines(fmax,vang(:),'r')

figure,plot(frs,vang1(1,:),'r'),grid on

hold on,plot(frs,vang1(2,:),'m')

hold on,plot(frs,vang1(3,:),'k')

hold on,plot(frs,vang1(4,:),'g')

hold on,plot(frs,vang1(5,:),'b'),title('Angles (in turns)')

plot\_lines(f0,vang1(:),'g')

plot\_lines(fmax,vang1(:),'r')

figure,plot(frs1,diff(vang1(1,:)),'r'),grid on

hold on,plot(frs1,diff(vang1(2,:)),'m')

hold on,plot(frs1,diff(vang1(3,:)),'k')

hold on,plot(frs1,diff(vang1(4,:)),'g')

hold on,plot(frs1,diff(vang1(5,:)),'b'),title('Angles differences')

plot\_lines(f0,diff(vang1(:))/4,'g')

plot\_lines(fmax,diff(vang1(:))/4,'r')

toc

## equi\_example

% equi\_example

addr='F:';

%-------------------------------

% simple

[bsd\_out,BSD\_tab\_out,stpar]=bsd\_equilego(addr,'BSD\_O3\_L',0,[108 109])

figure,plot(bsd\_out)

cont=cont\_gd(bsd\_out);

T=cont.t0+100; % approximate epoch

sour=pulsar\_3;

sour=new\_posfr(sour,T);

bsd\_corr=bsd\_dopp\_sd(bsd\_out,sour,-1); % doppler and spin-down correction

spec=bsd\_pows(bsd\_corr,2)

spmean=mean(spec)

SD=86164.09053083288;

xs=sour.f0+[-2 -1 0 1 2]/SD;

figure,semilogy(spec),plot\_lines(xs,spec,'g'),grid on

ylim([spmean\*0.5 spmean\*200])

pause(5)

xlim(sour.f0+[-1:2:1]\*30/SD)

pause(5)

xlim(sour.f0+[-1:2:1]\*3/SD)

%-------------------------------

% multiant for an optimezed analysis

clear bsds\_adr\_tab

bsds\_addr\_tab(1).addr=addr;

bsds\_addr\_tab(1).tab='BSD\_O3\_L';

bsds\_addr\_tab(2).addr=addr;

bsds\_addr\_tab(2).tab='BSD\_O3\_H';

bsds\_addr\_tab(3).addr=addr;

bsds\_addr\_tab(3).tab='BSD\_O3\_V';

[parants,Dt\_max]=check\_bsdsstr(bsds\_addr\_tab); % create parants (used in bsd\_multiant\_equi)

parants

wband=0.5; % MAcom is default

out=bsd\_multiant\_equi(bsds\_addr\_tab,sour,wband)

% MAcom describes the way of doing things; it is created by show\_multiant\_equi or show\_multiant\_equi\_full:

% otherwise the default are used

% Interactive setting for multiant

% previously create bsds\_addr\_tab and the sour (ex.: sour=pulsar\_3)

show\_multiant\_equi

% or show\_multiant\_equi\_full

%-------------------------------

% hardware injection management

% load the table cwinj\_O3 (or cwinj\_O2); it is in \Snag\projects\gw\sources\gw\_ps .

pulsar=read\_cwinj\_table(cwinj\_O3,7) % creates the pulsar 7

all\_hard\_inj % creates all the injected pulsars as single structures and

% as the full collection in a cell array named puls

# Code Blocks (“BSD snippets”)

In order to simplify the use of BSDs and establish standard procedures, **code blocks** have been introduced. They are a standard collection of scripts that should be

* copied in the local work folder
* renamed in order to not be confounded
* modified as needed
* used as such or put in another script

These blocks use the BSD global variables.

Some code blocks:

* bsd\_block\_0

% bsd\_block\_0

%

% defines the search parameters

%

% bsd\_block procedures: copy the script in the work folder

% with different name and modify

% do not modify original block

global bsd\_glob\_noplot bsd\_glob\_level

bsd\_glob\_set=reset\_bsd\_glob()

addr='I:';

ant='ligol';

runame='O2';

tim=1;

target=pulsar\_3;

SD=86164.09053083288;

* bsd\_block\_1

% bsd\_block\_1

%

% extracts data to analyze

% needs procedure bsd\_block\_0

% needs data extr\_half\_band

% produces bsd\_out,BSD\_tab\_out,stpar

%

% bsd\_block procedures: copy the script in the work folder

% with different name and modify

% do not modify original block

extr\_half\_band=0.05;

frdum=target.f0;

freqdum=[frdum-extr\_half\_band,frdum+extr\_half\_band];

BSD\_tab\_out=bsd\_tabout(addr,ant,runame,tim,freqdum);

ntab=length(BSD\_tab\_out.t\_ini);

T00=(BSD\_tab\_out.t\_ini(1)+BSD\_tab\_out.t\_fin(ntab))/2;

T0=BSD\_tab\_out.t\_ini(1);

Tobs=(BSD\_tab\_out.t\_fin(ntab)-BSD\_tab\_out.t\_ini(1))\*86400;

dsd0=1/Tobs^2; dsd=dsd0

target=new\_posfr(target,T0);

frdec=target.f0;

freq=[frdec-0.05,frdec+0.05];

freq1=[frdec-5.1/SD,frdec+5.1/SD];

mode=1; target

* bsd\_block\_noncohe\_frsd

% bsd\_block\_noncohe\_frsd

%

% non-coherent procedure to find fr and sd

% needs procedure bsd\_block\_1

% needs data sd1 and sd2

% produces bsd\_out,BSD\_tab\_out,stpar

%

% bsd\_block procedures: copy the script in the work folder

% with different name and modify

% do not modify original block

* bsd\_block\_fr\_sd\_lowres

% bsd\_block\_fr\_sd\_lowres

%

% low-resolution non-coherent procedure to find fr and sd

% needs procedure bsd\_block\_1

% needs data npiece, SDband, sd1 and sd2

% par search parameters (there is a default)

% .nl raw search reduction factor (def 5)

% .nh refined search enhancement factor (def 2)

% .refb refinement band (in natural units, def 5)

% .enl enlargement factor for bsd\_trim

% produces bsd\_out,BSD\_tab\_out,stpar

%

* bsd\_block\_sidpat

% bsd\_block\_sidpat

%

% produces and analyze the sidereal pattern

% needs bsd\_ftr

%

* bsd\_block\_sid\_ana

% bsd\_block\_sid\_ana

%

% sidereal analysis

% needs procedure bsd\_block\_1

% produces bsd\_out,BSD\_tab\_out,stpar

* bsd\_block\_modify\_target

% bsd\_block\_modify\_target

%

% simply modification of target

Examples of use:

% ex\_bsd\_zoomcand\_nosd

%

%

bsd\_block\_0

bsd\_block\_1

bsd\_block\_noncohe\_frsd

-----------------------

% check\_noncohe\_pos\_sid\_error

bsd\_block\_0

bsd\_block\_1

bsd\_glob\_noplot=1

target0=target;

da=0.1;

dd=0.1;

na=5;

nd=4;

snrout=zeros((2\*na+1),(2\*nd+1));

snrout2=snrout;

maxout=snrout;

as=(-na:na)\*da;

ds=(-nd:nd)\*dd;

ii=0;

for i = -na:na

ii=ii+1;

jj=0;

for j= -nd:nd

jj=jj+1;

target.a=target0.a+da\*i;

target.d=target0.d+dd\*j;

bsd\_block\_sid\_ana

snrout(ii,jj)=sid.datrat;

snrout2(ii,jj)=sid.gendatrat;

maxout(ii,jj)=max(sid.pow);

end

end

bsd\_glob\_noplot=0

figure,imagesc(maxout),title('sid.pat. max'),ylabel('r.a. error'),xlabel('dec. error')

figure,plot(as,maxout),grid on,xlabel('r.a. error'),title('sid.pat. max')

figure,plot(ds,maxout'),grid on,xlabel('dec. error'),title('sid.pat. max')

figure,imagesc(snrout),title('sid.pat. snr'),ylabel('r.a. error'),xlabel('dec. error')

figure,plot(as,snrout),grid on,xlabel('r.a. error'),title('sid.pat. snr')

figure,plot(ds,snrout'),grid on,xlabel('dec. error'),title('sid.pat. snr')

# **General job program**

Here is the general structure of a job program, as BSD\_JOB, DIRECT\_JOB and FOLLOWUP\_JOB:

[candout,job\_info,job\_summary,check]=GENERAL\_JOB(target,bsdin,mod)

Inside the program, proc\_info structure is created, starting from the type of job, the input parameters and the user choices represented in the mod structure.

The job\_info contains the produced information.

Appendix

## Debug & check

function [dopin,dopout]=check\_bsd\_doppler(in,doptabs)

% checks the dopper of a bsd

# Index

addobba\_notch 33

bsd\_5vec 36

bsd\_acc\_modif 26

bsd\_access\_str 26

bsd\_cand 29

bsd\_concsubband 17

bsd\_dayplot 18

bsd\_dopp\_sd 20

bsd\_frfilt 32

bsd\_fu 30

bsd\_holes 17

bsd\_hough 29

bsd\_interband 27

BSD\_JOB 29

bsd\_large\_band 27

bsd\_largerband 21

bsd\_lowres\_pows 37

bsd\_mock\_noise 20

bsd\_mode 29

bsd\_name\_piece 14

bsd\_orthoband 18

bsd\_par 16

bsd\_peakmap 23

bsd\_peakmap\_cell 24

bsd\_peakmap\_conv 24

bsd\_piece\_5vec 36

bsd\_pows 36

bsd\_real 20

bsd\_resamp 21

bsd\_resenh 32

bsd\_reshape 17

bsd\_seltab 15

bsd\_sid 37

bsd\_softinj 21

bsd\_specfilt 36

bsd\_subband 16

bsd\_tfclean 20

bsd\_tfstr\_add 22

bsd\_tfstr\_addwithnotch 22

bsd\_tfstr\_db 24

bsd\_tfstr\_db\_all 24

bsd\_timfr\_anabasic 23

bsd\_worm 36

bsd\_zeroholes 18

bsd\_zerointerv 17

check\_bsd\_doppler 55

check\_bsd\_table 15

check\_inj 37

conc\_bsd 15

crea\_BSD\_table 15

crea\_VPstr 14

cut\_bsd 15

DIRECT\_JOB 31

dopp\_corr\_residual 30

ex\_bsd\_5vec 52

ex\_bsd\_dedop 48

ex\_bsd\_dedop\_sing 49

ex\_bsd\_event 44

ex\_bsd\_modif 50

ex\_bsd\_sband 47

ex\_bsd\_softinj 50

file\_tab\_bsd 15

is\_bsd 18

load\_tab\_bsd 16

MCF

crea\_mc\_filter 43

mc\_5vec 43

mc\_filter 43

mc\_sn\_5vec 43

test\_mcf\_10 43

Other useful functions

complex\_clip 38

double\_threshold 39

extr\_band 38

logarithmize 38

stat\_nonzero 38

tfstr\_gd 38

sfdb09\_2bandgds\_BDS 13

show\_tfstr 23

show\_tfstr\_cell 23

VFS

crea\_spec\_filter 42

vfs\_create 40

vfs\_create\_1 40

vfs\_create\_2 40

vfs\_freq 41

vfs\_spindown 42

vfs\_subhet 41

vfs\_subhet\_pos 41