Wiki Syntax
Recent changes
Index
Search

S I G P R O C

The SIGPROC is a package which holds programs written to process, reduce and convert pulsar data. SIGPROC manual and package itself can be downloaded from here.

Software

Here are some SIGPROC programs:

`bandpass` - outputs the pass band from a filterbank file

```
usage: bandpass {filename} -{options}
N.B. filename - filterbank data file (def=stdin)
options:
 -d numdumps - number of dumps to average over (def=all)
 -t dumptime - number of seconds to average over (def=all)
```

`barycentre` - refer a datafile to a frame at rest wrt the solar system barycentre

```
usage: barycentre inputfile -{options} > outputfile
options:
 -mypolyco  - take user-defined polyco.bar file (def=create one)
 -verbose   - write out barycentre information to stderr (def=quiet)
  inputfile - the name of the filterbank/time series file
```

`best` - displays the best suspects from `seek`

```
usage: best <INFILE> -{options}
N.B. input file may be of the ".prd", ".top" or ".frq" variety
options:
 -v       - view output selectively
 -p       - produce postscript file output automatically
 -i       - zap integer harmonics only
 -t       - trace mode... show all harmonic relationships
 -f[fold] - read a specific harmonic fold (1-5;def=all)
 -s[smin] - set signal-to-noise threshold smin (def=8)
 -l[mini] - set minimum prd/frq to consider (ms;def=0)
 -h[maxi] - set maximum prd/frq to consider (ms;def=inf)
 -L[mini] - set minimum ac/dm to consider
 -H[maxi] - set maximum ac/dm to consider
 -F[fMHz] - set centre freq for DMsmear test (optional)
 -C[fMHz] - set channel band for DMsmear test (optional)
```

`chaninfo` - time series stats of 1-bit data

```
usage: chaninfo file (nblk) (cthr) (bthr)
options:
  file - name of raw .dat file
  nblk - (optional) blocks to read (def=all)
  cthr - (optional) threshold for bad channels (def=+/-0.2)
  bthr - (optional) threshold for bad blocks (def=3 sigma)
```

`csearch` - script to do a companion search on a dedispersed time series

```
usage: csearch filestem parstem
N.B. The script requires a .tim file with the dedispersed data and an "aclist" file
     which is an ASCII list of the trial ACs and ADOTs to use in the search.
```

`decimate` - reduce time and/or frequency resolution of filterbank data

```
usage: decimate {filename} -{options}
N.B. filename - filterbank data file (def=stdin)
options:
 -c numchans - number of channels to add (def=all)
 -t numsamps - number of time samples to add (def=none)
 -T numsamps - (alternative to -t) specify number of output time samples
 -n numbits  - specify output number of bits (def=input)
 -headerless - do not broadcast resulting header (def=broadcast)
```

`dedisperse` - form time series from filterbank data or profile from folded data

```
usage: dedisperse {filename} -{options}
N.B. filename - full name of the raw data file to be read (def=stdin)
options:
 -d dm2ddisp  - set DM value to dedisperse at (def=0.0)
 -b numbands  - set output number of sub-bands (def=1)
 -B num_bits  - set output number of bits (def=32)
 -o filename  - output file name (def=stdout)
 -c minvalue  - clip samples > minvalue*rms (def=noclip)
 -f reffreq   - dedisperse relative to refrf MHz (def=topofsubband)
```

```
  -F newfreq   - correct header value of centre frequency to newfreq MHz (def=header value)
  -n num_bins  - set number of bins if input is profile (def=input)
  -i filename  - read list of channels to ignore from a file (def=none)
  -p np1 np2   - add profile numbers np1 thru np2 if multiple WAPP dumps (def=all)
  -j Jyfactor  - multiply dedispersed data by Jyfactor to convert to Jy
  -J Jyf1 Jyf2 - multiply dedispersed data by Jyf1 and Jyf2 to convert to Jy (use only for two-polar
  -wappinvert  - invert WAPP channel order (when using LSB data) (def=USB)
  -wappoffset  - assume wapp fsky between two middle channels (for pre-52900 data ONLY)
  -swapout     - perform byte swapping on output data (def=native)
  -nobaseline  - don't subtract baseline from the data (def=subtract)
  -sumifs      - sum 2 IFs when creating the final profile (def=don't)
  -subzero     - subtract the zeroDM mean from each sample (def=don't)
  -headerless  - write out data without any header info
  -epn         - write profiles in EPN format (def=ASCII)
  -asciipol    - write profiles in ASCII format for polarization package
  -stream      - write profiles as ASCII streams with START/STOP boundaries
```

`depolyco` - resample a time series to either barycentric or pulsarcentric frames

```
usage: depolyco {timfile} {polycofile} -{options}
N.B. no polyco file implies barycentric correction to be applied
options:
   timfile    - the name of the time series to be read
   polycofile - the name of the polyco file for the pulsarcentric case
  -singlebyte - write output as unsigned characters (def=floats)
  -raj        - use different RA (J2000; hh:mm:ss.s) than header
  -decj       - use different DEC (J2000; dd:mm:ss.s) than header
  -verbose    - write out TEMPO information to stderr (def=quiet)
```

`fake` - produce fake filterbank format data for testing downstream code

```
usage: fake -{options}
options:
 -period p   - period of fake pulsar in ms (def=random)
 -width w    - pulse width in percent (def=4)
 -snrpeak s  - signal-to-noise ratio of single pulse (def=1.0)
 -dm d       - dispersion measure of fake pulsar (def=random)
 -nbits b    - number of bits per sample (def=4)
 -nchans n   - number of filterbank channels (def=128)
 -tsamp t    - sampling time in us (def=80)
 -tobs t     - observation time in s (def=10)
 -tstart t   - MJD time stamp of first sample (def=50000.0)
 -nifs n     - number of IFs (def=1)
 -fch1 f     - frequency of channel 1 in MHz (def=433.968)
 -foff f     - channel bandwidth in MHz (def=0.062)
 -seed s     - seed for Numerical Recipes ran1 (def=seconds since midnight)
 -nosmear    - do not add in dispersion/sampling smearing (def=add)
 -swapout    - perform byte swapping on output data (def=native)
 -evenodd    - even channels=1 odd channels=0 (def=noise+signal)
 -headerless - do not write header info at start of file (def=header)
binary options:
 -binary     - create binary system
 -bper       - orbital period in hours (def=10.0)
 -becc       - eccentricity (def=0.0, circular)
 -binc:      - inclination in degrees (def=90.0)
 -bomega     - longitude of periastron in degrees (def= 0.0)
 -bphase     - starting orbital phase (number between 0 and 1, def=0.0)
 -bpmass     - pulsar mass in solar units (def=1.4))
 -bcmass     - companion mass in solar units (def=5.0)
```

`filterbank` - convert raw pulsar-machine data to filterbank format

```
usage: filterbank <rawdatafile1> .... <rawdatafileN> -{options}
N.B. rawdatafile - raw data file (recognized machines: WAPP, PSPM, OOTY)
options:
 -o filename - output file containing filterbank data (def=stdout)
 -s skiptime - skip the first skiptime (s) of data (def=0.0)
 -r readtime - read readtime (s) of data (def=all)
 -i IFstream - write IFstream (IFstream=1,2,3,4)
 -n nbits    - write n-bit numbers (def=input format)
 -c minvalue - clip DM=0 samples > mean+minvalue*sigma (def=noclip)
 -swapout    - perform byte swapping on output data (def=native)
 -floats     - write floating-point numbers (equal to -n 32)
 -sumifs     - sum IFs 1+2 to form total-power data
 -headerfile - write header parameters to an ASCII file (head)
 -headeronly - write ONLY binary header parameters
options for correlator (currently WAPP) data:
 -hammingv   - apply Hamming window before FFT (def=nowindow)
 -hanning    - apply Hanning window before FFT (def=nowindow)
 -novanvleck - don't do van Vleck correction before FFT (def=doit)
 -invert     - invert the band after FFT (def=noinversion)
 -zerolag    - write just the zero-lag value for each IF
 -rawcfs     - write raw correlation functions (novanvleck)
 -corcfs     - write corrected correlation functions (vanvleck)
```

`flux` - TBD

`fold` - fold filterbank channels/time series data

```
usage: fold {filename} -{options}
N.B. filename - full name of the raw data file to be read (def=stdin)
options:
 -o out_file - output file for pulse profile data (def=stdout)
 -p fold_prd - period to fold (ms) or polyco file (def=polyco.dat)
 -a accelern - fold using constant acceleration (def=0 m/s/s)
 -f p_factor - multiply the period by p_factor (def=1.0)
 -m m_factor - output multiple profiles (STREAM only; def=1)
 -n num_bins - number of bins in folded profile(s) (def=window/tsamp)
 -d time/num - dump profiles every time s or num pulses (def=nodumps)
 -t samptime - hard-wire the sampling time (us) (def=header)
 -l phaseval - phase value (turns) of left edge of pulse (def=0.0)
 -r phaseval - phase value (turns) of right edge of pulse (def=1.0)
 -j Jyfactor - multiply all profiles by Jyfactor to convert to Jy
 -b baseline - subtract baseline from all profiles (def=autobase)
 -dt timeoff - add a time offset in seconds to tstart (def=0.0)
 -sk skiptim - skip the first skiptim s before folding (def=0.0)
 -re readtim - read and fold only readtim s of data (def=ALL)
 -ascii      - write profiles as ASCII numbers (this is the default)
 -epn        - write profiles in EPN format (def=ASCII)
 -acc        - write out accumulated profiles (def=subints)
 -bin        - write profiles in SIGPROC binary format (def=ASCII)
 -sub subint - shorthand for -nobaseline -stream -d subint.0
 -psrfits    - write profiles in PSRFITS format (def=ASCII)
 -totalpower - sum polarizations 1+2 before writing (def=nosumming)
 -asciipol   - write profiles in JMCs ASCII format for polarization
 -stream     - write profiles as ASCII streams with START/STOP bounds
 -nobaseline - don't subtract baseline from profiles (def=subtract)
```

`foldsignals` - script to dedisperse and fold signals from SEEK

```
usage: foldsignals stem -{options}
options:
 -nbands n   - number of sub-bands in output plot (def=8)
 -ntrials n  - number of trial periods to try (def=64)
 -nsubints n - number of sub-integrations in output plot (def=8)
 -snmin s    - miniumum signal-to-noise ratio (def=8)
 -nbins n    - number of bins in output profile (def=P/tsamp)
 -ncand n    - process only candidate number n (def=all)
 -listonly   - just list the candidates from best
```

`getpulse` - make and/or plot a time series from dedispersed file

```
usage: getpulse {filename} -{options}
N.B. filename - dedispersed data file
options:
 -t time   - time (in seconds) on which to center time series (REQUIRED)
 -w width  - width (in seconds) of time series to plot (def=1)
 -c fchan  - to only output frequency channel fchan (def=all)
 -i ifchan - to only output IF channel ifchan (def=all)
 -p pgdev  - pgplot device (def=/xs)
 -numerate - to precede each dump with sample number (def=time)
 -noindex  - do not precede each dump with time/number
 -plot     - PGPLOT the results
```

`grey` - program to greyplot EPN data

```
usage: plotg [filename] <options>
N.B. Input file must be in EPN format!
options:
 -r - set start record number (def=1)
 -s - set min=0 & max=n*rms (def=autoscale)
 -c - centres profile to first record (optional)
```

`header` - program to display header information

```
usage: header {filename} -{options}
N.B. filename - filterbank data file (def=stdin)
options:
 -telescope   - return telescope name
 -machine     - return datataking machine name
 -source_name - return source name
 -fch1        - return frequency of channel 1 in MHz
 -foff        - return channel bandwidth in MHz
 -nchans      - return number of channels
 -tstart      - return time stamp of first sample (MJD)
 -tsamp       - return sample time (us)
 -nbits       - return number of bits per sample
 -nifs        - return number of IF channels
 -headersize  - return header size in bytes
 -datasize    - return data size in bytes if known
 -nsamples    - return number of samples if known
 -tobs        - return length of observation if known (s)
```

`hunt` - script to dedisperse and search data using seek over a given DM range

```
usage: hunt filestem (option)
N.B. The script requires a filterbank file with the raw data and a "dmlist" file
```

```
      which is an ASCII list of the trial DMs to use in the search.
```

`makedmlist` - program

```
usage: makedmlist name_of_filterbank_file
N.B. DM minimum and maximum ranges are optional (0-100 is default).
options:
 -dmmin - minimum value of DM
 -dmmax - maximum value of DM
```

`mask` - produces a spectral mask for use by `seek`

```
usage: mask <SPECTRUM_FILE> -{options}
N.B. The spectrum file is produced by running find -s
     This program is usually run on fold1.spc
     Options given as: mask fold1.spc -t10
options:
 -f       - write the spectral mask to file "mask.out"
 -l[f_hz] - lowest frequency to consider (def=0)
 -h[f_hz] - highest frequency to consider (def=Nyqst)
 -t[ampl] - amplitude threshold to mask
```

`monitor` - Tcl/Tk widget to check and display log output from programs

`pgplotter` - simple tool to plot SIGPROC streamed output with PGPLOT

`plotpulses` - PGPLOT results of single pulse search

```
usage: plotpulses filestem1 .... filestemn -{options}
options:
 -minsn s1 - minimum s/n to display (def=5)
 -maxsn s2 - maximum s/n to display (def=all)
 -pgdev    - pgplot device (def=/xs)
 -wmin w1  - minimum width (in ms) to display (def=all)
 -wmax w2  - maximum width (in ms) to display (def=all)
 -dmmin d1 - minimum DM (def=all)
 -dmmax d2 - maximum DM (def=all)
 -tmin t1  - minimum time (seconds) to display (def=all)
 -tmax t2  - maximum time (seconds) to display (def=all)
 -dmplot d - 1 to plot DM and 2 to plot DM channel (def=1)
 -nmax n   - maximum number of events to plot per DM channel (def=all)
 -allbeams - will do a DM-t stack for all beams in the current directory
```

`polyco` - a script to run TEMPO to generate a polyco.dat file

```
usage: polyco psrname -{options}
N.B. psrname - name of the pulsar as it appears in tztot.dat
options:
 -freq f   - frequency in MHz (def=1410 MHz)
 -nspan n  - span of each polyco set in minutes (def=15 min)
 -ncoeff n - number of coefficients in each polyco set (def=9)
 -maxha h  - maximum hour angle (def=2 hours)
 -mjd m    - mjd to calculate for (def=today)
 -mjds s   - starting mjd (def=today)
 -mjdf f   - finishing mjd (def=today)
 -par p    - specify parfile (def=tzpar area)
 -site s   - specify site code or alias (def=Arecibo)
```

`polyco2period` - returns period (sec) given mjd and polyco file

```
usage: polyco2period mjd -{options}
options:
  mjd - MJD date
 -p polyco file name
```

`postproc` - TBD

`profile` - produce ASCII or pseudo grey-scale displays of folded data

```
usage: profile {filename} -{options}
N.B. filename - profile file (def=stdin)
options:
 -p fraction - set max value to fraction of peak (def=1.0)
 -frequency  - label grey-scale profiles in frequency (def=time)
```

`quicklook` - script to examine the data

```
usage: quicklook <filename> -{options}
options:
 -read time   - read and process only time (s) of data
 -skip time   - skip the first time (s) of data
 -addc nchans - add nchans chans together before dedispersion (def=none)
 -addt nsamps - add nsamps samples together before dedispersion (def=none)
 -nsints   n  - specify number of time sub-integrations (def=8)
 -nbands   n  - specify number of frequency sub-bands (def=8)
 -period p    - fold data at constant period (ms) (def=polyco)
 -dm dmvalue  - dedisperse using dmvalue (pc cm-3) (def=polyco)
 -clip value  - clip samples that deviate more than value*rms (def=noclipping)
```

```
  -nbins n     - fold data using n bins (def=128)
  -left phase  - specify left-hand phase window (def=0.0)
  -right phase - specify right-hand phase window (def=0.0)
  -singlepulse - set number of subints to be the number of pulses
  -psr name    - fix source name for running TEMPO (def=filestem)
  -mypolyco    - use an existing polyco file (def=make one)
  -fakescale   - plot channels/bands using fake grey scale (def=real)
  -clean       - remove large files before and after (def=keep them)
  -wipe        - remove large files beforehand (def=keep them)
  -ext string  - add a file extension string to output files (def=none)
```

`reader` - look at filterbank data in ASCII format

```
usage: reader {filename} -{options}
N.B. filename - filterbank data file (def=stdin)
options:
 -c        - output only frequency channel c (1...nchans) (def=all)
 -i        - output only IF channel i (1...nifs) (def=all)
 -numerate - precede each dump with sample number (def=time)
 -noindex  - do not precede each dump with number/time
 -stream   - produce a stream of numbers with START/STOP boundaries
```

`seek` - searches for periodic and/or transient signals in a noisy time series

```
usage: seek <INFILE> -{options}
N.B. The input file may be a time series, or a set ofFourier coefficients.
     The file extension MUST, however, be either ".tim" ".ser" ".dis" or ".fft"
     In the latter case, the FFT stage is skipped.
options:
 -A       - append output ASCII files
 -s       - dump spectra to ".spc" files
 -q       - quiet mode - all messages > INFILE.log
 -pmzap   - calls the PM survey routine pm_zapbrd
 -mmzap   - calls the PM survey routine mm_zapbrd
 -pulse   - calls Maura single pulse search
 -pzero   - pads out data with zeros (def=Gaussian)
 -maxft   - report max length of Fourier transform
 -nofft   - turns off FFT search
 -fftw    - uses FFTW instead of SINGLETON routine
 -hsums   - show harmonic summing used
 -recon   - calculate and report reconstructed S/N
 -submn   - mean subtraction (old method) to whiten spectrum
 -submd   - median subtraction to whiten spectrum
 -head    - adds header info to output .prd files
 -m[file] - mask birdies from file (def="mask")
 -z[file] - zap birdies from file (def="birdies")
 -b[freq] - zap 10-sig+ spikes < freq (def=100 Hz)
 -c[cfac] - add every cfac samples before FFT
 -a[accn] - re-sample at constant accn (m/s/s)
 -d[adot] - re-sample at constant adot (cm/s/s/s)
 -D[dmvl] - change header DM to be dmvl
 -t[tlen] - fix transform length to 2**tlen
 -i[tsec] - ignore tsec seconds of data on reading
 -p[pmax] - set maximum period of seach (def=9.999s)
 -T[spth] - set single-pulse search threshold (sigma)
 -n[nmax] - maximum number of single-pulse candidates per DM channel
 -w[smax] - number of times to smooth time series for single-pulse search
```

`spec` - displays spectrum files produces by `find`

```
usage: spec <SPECTRUM_FILE> -{options}
N.B. The spectrum file is produced by running find -s
     This program is usually run on fold1.spc
options:
 -s       - run stats mode
 -d       - dump ASCII version of spectrum to file "dump"
 -P       - plot powers (def=amplitudes)
 -l[f_hz] - lowest frequency to consider (def=0)
 -h[f_hz] - highest frequency to consider (def=Nyqst)
 -f[fold] - give fold number instead of filename
 -n[nbin] - number of bins (histograms) in stats mode
 -F[f_hz] - mark frequency and harmonics on plot
```

`splice` - joins multiple filterbank files

```
usage: splice filename1...filenameN > splice.fil
N.B. filename - filterbank data file (def=stdin)
```

`splice_timeseries` - joins multiple timeseries files

```
usage: splice_timeseries filename1...filenameN > splice.tim
N.B. filename - timeseries data file (def=stdin)
```

`step` - TBD

`tune` - fine tune a period in a time series by stacking sub-integration

```
usage: tune {filename} -{options}
```

```
N.B. filename - full name of the raw data file to be read (def=stdin)
options:
 -p fold_prd - center period to fold (ms) or polyco file (def=polyco.dat)
 -a accelern - center acceleration to fold (def=0 m/s/s)
 -f p_factor - multiply the period by p_factor (def=1.0)
 -n num_bins - max number of bins in folded profile(s) (def=64)
 -t samptime - hard-wire the sampling time (us) (def=header)
 -sub subint - Number of subints to use (def=128)
 -quikgray   - plot data using pdm-style quikgray code (def=pggray)
 -useaccn    - Do an acceleration search *Experimental* (def=don't)
 -usejerk    - Do an acceleration search *Experimental* (def=don't)
 -pf pfactor - divide the period search range by this number (def=1)
 -af afactor - divide the accel search range by this number (def=1)
 -jf jfactor - divide the jerk search range by this number (def=1)
 -format fmt - set the output format, standard PGPLOT formats (def=/xserv)
 -jreaper f  - write out ascii based output to file f (def=don't)
 -bestfile f - read the 'best' summery file for candidate info (def=don't)
```

Data reduction

Here is the example how from time series file, fold and convert the data into an epn file.

First the TOA file for the pulsar (polyco.dat) should be generated using TEMPO package. You can do it by running command:

```
tempo -z pulsar_name.tz -f pulsar_name.par
```

This will generate a polyco.dat file which will be used (unless you specify otherwise) to fold the data.

To fold the data and write it to epn format with `fold` program, you can use command like this:

```
fold -d 1 -p 3.0 filename -epn > filename.epn
```