Description of PowerFlux 2 algorithms and implementation (draft)

LIGO-T1000272-v1

Vladimir Dergachev

May 25, 2010

Contents

List of Figures

1 Introduction

PowerFlux is a program for estimating the flux of monochromatic gravitational waves from a particular source in the sky. It uses short Fourier transform files (SFTs) of 30 minute duration (also called "coherence time") as input data. Because of this, PowerFlux is much less sensitive to variations in phase than coherent methods (for example ComputeFStatistic). The result is a tradeoff of sensitivity for drastic reduction of search space, allowing for full-sky full-bandwidth searches to be completed in reasonable time.

This algorithm has been previously described in [1, 2, 3].

The original PowerFlux code was designed for efficient computation of power sums in 0.25 Hz band while iterating over sky templates, spindowns and polarizations. A Feldman-Cousins algorithm [4] was then used to obtain upper limits. A number of auxiliar information was collected as well, in particular signal-to-noise ratios, locations of outliers and values of Kolmogorov-Smirnov tests to verify Gaussianity of underlying data.

The all-sky search of full S5 dataset which spanned 2 years of data presented a number of challenges:

- The spindown step used in the search had to be decreased to 3×10^{-11} Hz/s from 5×10^{-10} Hz/s used in previous searches. This greatly increased the computing time needed to cover the same range of parameters.
- The larger dataset strained memory limits of available computing clusters. At the moment the largest cluster is ATLAS which makes available 2 GB of memory per node - a soft limit as actual quad-core machines have 8 GB of memory which is shared between 4 nodes.
- For a good portion of frequency range full S5 dataset is the most sensitive to date and is expected to be superseded only when advanced LIGO or Virgo detectors come into operation. This presents a problem as we cannot rely on more sensitive data to confirm or reject potential candidates.

These issues were met by rewriting PowerFlux analysis pipeline to accommodate new search methods and increase efficiency. The existing startup code - including sky grid generation, SFT input-output and software injection engine - was reused with only minimal changes to accommodate new pipeline. Thus both original (PowerFlux) and new (PowerFlux2) codes can

be compiled from the same common code base and tested using the same software injection code.

Both executables can run from the same configuration file and the options not applicable to the particular code are ignored.

The following features are new in PowerFlux2:

- Partial power sum cache provides 5-10x speedup when running in single bin mode.
- The ability to produce upper limits for subsets of the entire data broken down by time and interferometers, as well as their combinations. Thus outliers for individual interferometers and using all interferometers together are obtained from the same instance.
- Utilization of vector processing instructions of modern CPUs.
- Extensibility to new methods of computation power sums.
- Higher sensitivity (but slower) *loosely coherent* mode [5] of computing power sums.

2 Power sum caching

The most significant change in PowerFlux is the introduction of partial power sum cache. The idea is as follows: suppose we are analyzing a stretch of data several month long. Then the minimal change in spindown value that the search can tolerate is determined by this large timebase.

Consider now a partition of the power sum over the entire dataset into stretches 2 week long. Each stretch can tolerate significantly larger spindown step. If we precompute the partial power sums for each stretch we can quickly conclude the entire computation by picking the appropriate partial power sum altered by some constant frequency shift for each finer spindown value.

This approach can be further elaborated by noting that the other large source of frequency shift is Doppler effect from Earth orbital velocity vector which also stays almost constant on the scale of a few weeks. Thus additional savings can be realized by extending the same principle to groups of nearby sky templates.

The actual implementation groups all of these concepts together.

First, we compute power sums for groups of sky templates for which we can assume constant amplitude response (we will call such a group a "patch").

Secondly, the only thing that matters for computation of power sums are per-SFT frequency shift and, for the loosely coherent search, the relative phase adjustment.

Then an associative cache that uses sequences of frequency shifts as key will provide a model-independent way to accelerate the code and take advantage of degeneracy between signal parameters on short timescales.

Several nuances result in improved performance:

- In practice, of course, the frequency shifts are real numbers. To use them as a key one should round them to the nearest point on a frequency grid. For single bin PowerFlux mode this means rounding to the nearest frequency bin.
- Two sequences of shifts that are different by the same constant in every term can be easily derived from the same power sum if it had enough extra points to accommodate the offset.
- It is best to group SFTs by the magnitude of their frequency shift. This way they are more likely to differ by only a constant shift from previously computed group.

The result of all these improvements is that a single spindown run in single-bin PowerFlux mode usually sees cache efficiency on the order of 90%, while a multiple spindown run can achieve efficiency of 95%. The cached sums still need to be accumulated and analyzed, so the overall gain in speed is between 5x (for single spindown run) to 10x (for multiple spindowns).

3 PowerFlux2 architecture

3.1 Data structures overview

Like its predecessor, PowerFlux2 is structured as a computational engine where a library of functions operates on data structures that carry information throughout the pipeline.

The following data structures are particularly important:

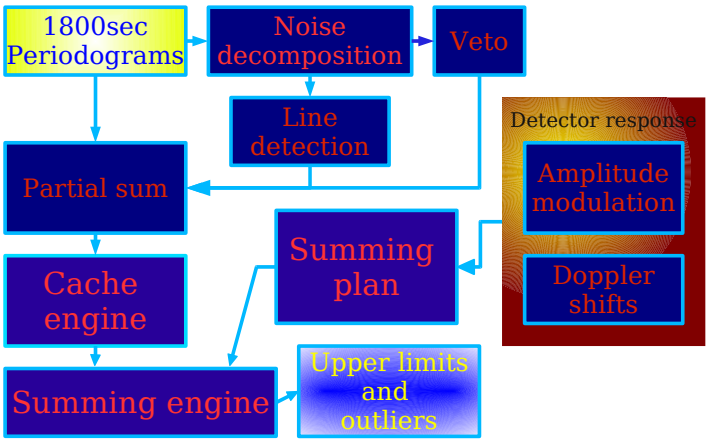

Figure 1: Flowchart of PowerFlux code

- `PARTIAL_POWER_SUM` carries the results of accumulating power from a subset of available SFTs. Instead of specializing to particular polarizations (as has been done in earlier version of PowerFlux) we carry the coefficients of the polynomials (quadratic in power and quartic in weight) that depend on polarization coefficients.
- `SEGMENT_INFO` is filled in with information on a single SFT to be included into `PARTIAL_POWER_SUM`
- `SIMPLE_CACHE` keeps the cache of previously computed partial power sums.
- `POWER_SUM` contains information on particular template (sky location, spindown and sub-bin frequency shift) and computed partial power sum.
- `SUMMING_CONTEXT` carries the parameters that define which particular flavour of power sum is being computed as well as information local to a particular computational thread. In the current code there is one such structure for each allocated thread.
- `ALIGNMENT_COEFFS` specifies a particular polarization and contains precomputed coefficients ready to convert a `PARTIAL_POWER_SUM` into a weighted

sum for computation of upper limits.

- `POINT_STATS` is used to store computed upper limit, signal-to-noise ratio and other statistics for a particular polarization, sky location, spindown and sub-bin frequency shift.

- `POWER_SUM_STATS` carries aggregate information for a particular power sum or a set of templates.

- `EXTREME_INFO` carries aggregate information for the entire computation. The skymap members are `NULL` by default which saves memory in batched analysis.

3.2 Partial power sum

The figure 2 shows the declaration of the `PARTIAL_POWER_SUM` structure. The weighted power sum depends quadratically on amplitude modulation, while the sum of weights depends on it in the fourth degree. The individual components are labeled `pp`, `pc` and `cc` for the quadratic case and `pppp`, `pppc` and so on for the quartic case.

There are two sets of fields accumulating weights - the constant fields hold overall weight, while weight arrays are used only of line avoidance is being performed.

3.3 SFT information

The array of structures `SEGMENT_INFO` 3 is used as a summing plan for the uncached power sum functions. It carries the information on which SFTs with what frequency shifts and amplitude response coefficients will be summed. The actual functions are thus isolated from a specific signal model that is used in the power sum accumulation function.

The code implementing partial power sum cache ignores amplitude response coefficients, so they should be kept fixed between calls to reset the cache.

3.4 Simple cache

To speedup computation PowerFlux keeps an associative cache 4 of computed partial power sums. The `id` member identifies which particular pipeline

```
typedef struct {
        int type;
        int nbins;

        REAL c_weight_pppp;
        REAL c_weight_pppc;
        REAL c_weight_ppcc;
        REAL c_weight_pccc;
        REAL c_weight_cccc;

        REAL *weight_pppp;
        REAL *weight_pppc;
        REAL *weight_ppcc;
        REAL *weight_pccc;
        REAL *weight_cccc;

        /* power sums - plus^2, plus*cross and cross^2*/
        REAL *power_pp;
        REAL *power_pc;
        REAL *power_cc;

        int offset; /* this is the index of the bin with index 0 in the output (i.e. fir

        int weight_arrays_non_zero;
        int collapsed_weight_arrays;
        } SUFFIX(PARTIAL_POWER_SUM);
```

Figure 2: PARTIAL_POWER_SUM

```
typedef struct {
/* fields below are filled in when computing power */

/* amplitude response factors to use in power sum - note these are kept constant through
float f_plus;
float f_cross;

/* bin shift to apply, this is in units of 1/coherence_time - as opposed to power sums *
double bin_shift;
double diff_bin_shift; /* linear component, drift from one frequency bin to another */

/* fields below are filled in when locating segments */

/* for convenience, same as datasets[dataset].gps[segment] */
double gps;
double detector_velocity[3]; /* also from datasets[dataset] */
double coherence_time;


/* segment coordinates */
int dataset;
int segment;
int index; /* arbitrary index for use by power_sum accumulation code, typically for refe
} SEGMENT_INFO;
```

Figure 3: SEGMENT_INFO

flavour created it.

During execution the cache collects statistics on its performance. Besides basic `hits` and `misses` counters, a cache fault can be caused by an `overwrite` when key collision occurs or `large_shift` when we are requesting to compute a power sum with a constant frequency offset too large to be accommodated by built-in window.

The cache structure is reset for different sets of templates ("patches") and different SFT sets. The SFT count it expects is kept in `segment_count` variable. The actual cache keys are the sequences of `SEGMENT_INFO` structures kept in `si` member. Their hashes are stored in the `key` field of the structure.

```
typedef struct {
long id;

/* statistics */
long hits;
long misses;
long overwrites;
long large_shifts;
int max_size;

/* cache contents */
int segment_count;
int size;
int free;
int *key;
SEGMENT_INFO **si;
PARTIAL_POWER_SUM_F **pps;
} SIMPLE_CACHE;
```

Figure 4: `SIMPLE_CACHE`

3.5 Power sums

The `POWER_SUM` 5 data structure wraps `PARTIAL_POWER_SUM` with information sufficient to calculation frequency shift and amplitude response passed to partial power sum functions. It thus acts as a template. The partial power sum

will hold the accumulated data, and `min_gps` and `max_gps` members record the actual sampled timebase.

An array of power sum structures is passed to power sum accumulation function and all the templates specified by the array are computed simultaneously and share the same cache.

```
typedef struct S_POWER_SUM {
float freq_shift; /* additional shift e.g. for half-bin sampling */
float spindown;
float ra;
float dec;
float patch_ra;
float patch_dec;

double min_gps;
double max_gps;

float e[26];
float patch_e[26];

int skyband;

PARTIAL_POWER_SUM_F *pps;
} POWER_SUM;
```

Figure 5: `POWER_SUM`

3.6 Summing context

The summing context keeps thread specific information as well as methods defining which particular computation is being done.

The figure 6 shows the declaration of the structure. The method `get_uncached_power_sum` performs the actual computation of partial power sum and defines the search being performed. The accumulate methods implement the cache of previously computed power sums as well as calling sequence that makes most use of it. The same method can be shared between different search codes, but usually needs tuning to make the most of the cache while still accurately computing the power sums.

```
typedef struct S_SUMMING_CONTEXT {
void (*get_uncached_power_sum)(struct S_SUMMING_CONTEXT  *ctx, SEGMENT_INFO *si, int cou
void (*accumulate_power_sum_cached)(struct S_SUMMING_CONTEXT  *ctx, SEGMENT_INFO *si, in
void (*accumulate_power_sums)(struct S_SUMMING_CONTEXT *ctx, struct S_POWER_SUM *ps, int

int cache_granularity;
double inv_cache_granularity;
double half_inv_cache_granularity;

int diff_shift_granularity;
double inv_diff_shift_granularity;
double half_inv_diff_shift_granularity;

int sidereal_group_count; /* group sfts falling on similar times of the day in this many
double summing_step; /* process SFTs in blocks of this many seconds each */
int time_group_count; /* group SFTs by their GPS time within a block into this many grou

void *cache;
void (*free_cache)(struct S_SUMMING_CONTEXT *ctx);
void (*print_cache_stats)(struct S_SUMMING_CONTEXT *ctx);
void (*reset_cache)(struct S_SUMMING_CONTEXT *ctx, int segment_count, int template_count

void *patch_private_data;

/* dynamic parameters */
int loose_first_half_count;

} SUMMING_CONTEXT;
```

Figure 6: SUMMING_CONTEXT

The most important tuning parameter is `cache_granularity` which controls the sub-frequency bin tolerance of the cache for the purpose of computing frequency shifts. The parameter `diff_shift_granularity` performs the same function for frequency-dependent Doppler shift variation that loosely coherent search is sensitive to.

To make the most use of the cache the SFTs are partitioned in blocks of `summing_step` seconds and each block is further partitioned into `sidereal_group_count` groups by the magnitude of the frequency shift - in most cases it mostly depends on the hour of day.

3.7 Alignment coefficients

The `ALIGNMENT_COEFFS` 7 structure holds precomputed amplitude response coefficients, which make easy to compute numerator and denominator of weighted power sum by simply taking a dot product of (pp, pc, cc) and $(pppp, pppc, ppcc, pccc, cccc)$ members of the alignment structure with accumulated `PARTIAL_POWER_SUM`.

An array of alignment coefficients is allocated - one for each sampled polarization. A circular polarization is always added, so the total number of polarizations sampled is `npsi`·`niota`+1.

```
typedef struct {
float iota;
float psi;

float pp;
float pc;
float cc;

float pppp;
float pppc;
float ppcc;
float pccc;
float cccc;
} ALIGNMENT_COEFFS;
```

Figure 7: `ALIGNMENT_COEFFS`

3.8 Point statistics

The point statistics structure 8 contains upper limit and signal-to-noise ratio estimates for a particular template and particular polarization, as well as many other statistic parameters of various usefulness. The `iota` and `psi` members specify particular polarization this data was computed for. The four last fields are used to convey complete information on point for which the values were achieved - which is the maximum of upper limit and signal-to-noise ratio (both maximums are achieved for the same frequency bin).

`ks_value` has the value of Kolmogorov-Smirnov statistic when `statistics-function` is set to `sorted`. Otherwise the code fills in `m1_neg`, `m3_neg` and `m4` members which are the negative parts of first and third moments and a fourth moment. These are used to check for Gaussianity of computed power sums.

`weight_loss_fraction` describes how much underlying data was discarded due to line veto.

3.9 Accumulated statistics

`POWER_SUM_STATS` 9 holds the accumulated statistics from `POINT_STATS` structures computed for many polarizations and templates.

The inclusion of complete `POINT_STATS` structures for highest upper limit and signal-to-noise ratio points allows to identify where they were achieved.

3.10 Extremal points

The structure `EXTREME_INFO` 10 carries information about points which achieve maxima and minima of various statistics over different sets of templates. The skymap members contain maxima over spindowns, polarizations and frequencies. They can be set to `NULL` to conserve memory. The `band_info` members carry information about extremal points in each sky band.

There is one extreme info structure for each chunk of SFT data under analysis. The member `first_chunk` and `last_chunk` specify the start and end of the contiguous set of SFTs to analyze. `veto_num` indicates whether we are using SFTs from a particular detector only, or all detectors simultaneously (-1).

```
typedef struct {
double iota;
double psi;

double ul;
double ll;
double centroid;
double snr;

double M;
double S;
double ks_value;
double m1_neg;
double m3_neg;
double m4;
double max_weight;
double weight_loss_fraction;

int ks_count;

int bin;

/* the following fields are for convenience and are filled in by outside code based on v
double frequency;
double spindown;
double ra;
double dec;

} POINT_STATS;
```

Figure 8: POINT_STATS

```
typedef struct {
POINT_STATS highest_ul;
POINT_STATS highest_snr;
POINT_STATS highest_ks;
POINT_STATS highest_M;
POINT_STATS highest_S;
POINT_STATS highest_circ_ul;
double max_weight_loss_fraction;
double max_weight;
double min_weight;
double max_m1_neg;
double min_m1_neg;
double max_m3_neg;
double min_m3_neg;
double max_m4;
double min_m4;
int ntemplates;
} POWER_SUM_STATS;
```

Figure 9: POWER_SUM_STATS

```
typedef struct {
char *name;

float *ul_skymap;
float *circ_ul_skymap;
float *snr_skymap;
float *ul_freq_skymap;
float *circ_ul_freq_skymap;
float *snr_freq_skymap;
float *snr_ul_skymap;
float *max_weight_skymap;
float *min_weight_skymap;
float *weight_loss_fraction_skymap;
float *ks_skymap;

POWER_SUM_STATS *band_info;
int *band_valid_count;
int *band_masked_count;

/* convenience info for keeping track of which ei is which */
int first_chunk;
int last_chunk;
int veto_num;
} EXTREME_INFO;
```

Figure 10: EXTREME_INFO

4 Single bin semi-coherent statistic

The single-bin semi-coherent summing mode is turned on by specifying `averaging-mode=one`. This implements conventional weighted power sum algorithm, with several optimizations that increase speed by up to a factor of 10 when iterating over many small spindown steps. The "single-bin" refers to the use of a single power bin from each Hann windowed SFT. It is possible to create a variation of the same algorithm that uses several SFT bins for more accurate power estimation (called "matched filter" mode), however the matched filter employed for that purpose is computationally expensive and the results are more sensitive to sub-bin frequency shift.

4.1 Computation of partial power sums

The uncached partial power sum function computes the partial power sum taking into account amplitude modulation and SFT noise level. The weight estimation can be performed either by using the `TMedians` as in the first version of PowerFlux, or by computing variance of SFT bins actually used. This results in small improvement in sensitivity. The non-robustness of variance is actually useful as it deemphasizes SFTs with large spikes. Which of the two methods is used is determined by `tmedian-noise-level` parameter.

If line veto is on, the algorithm avoids lines by subtracting out the contribution of affected frequency bins.

Two variants of the function has been written - a regular C implementation and an optimized version using explicit calls to functions implementing vector based arithmetic.

4.2 Caching of partial power sums

The cached accumulation function implements a hash table that holds previously computed power sums. This assumes that amplitude modulation constants are the same for all templates being processed, therefore, the cache is emptied at the start of each patch with different amplitude modulation constants.

The key is based on the well-known modulo arithmetic algorithm applied to frequency bin shifts. The value of the first bin shift is subtracted out as translation by integer number of frequency bin can be easily accommodated by computing slightly more frequency bins than necessary.

4.3 Power sum accumulation

The power sum accumulation function computes the power sums in blocks of `summing_step` seconds (by default 10 days). Each block is partitioned into SFT groups by the magnitude of the frequency shifts. The total number of groups is governed by `sidereal-group-count` parameter. Note that the groups do not have to be equal in length, in fact computation will be more efficient in the case of one large group.

5 Loosely coherent statistic

The loosely coherent method is activated by specifying `averaging-mode=loose_single_bin`. This turns on higher sensitivity, though slower, mode meant for investigation of small portions of the sky.

The code is contained in files `single_bin_loosely_coherent_sum.c` and `single_bin_loosely_coherent_sum.h` which define uncached partial power sum function as well as corresponding accumulation methods.

5.1 Computation of partial power sums

The uncached partial power sum function assumes that its input is split into two sets of SFTs (which may coincide). The number of SFTs in the first step is governed by parameter `loose_first_half_count` in the summing context. This function returns the double sum:

$$P_k = \sum_{i,j} b^*_{kj} K_{ji} a_{ki}$$

where a_{ki} are phase corrected SFT bins from the first set, index k differentiates different SFT bins, while i corresponds to time. b_{ki} denotes the phase corrected SFT bins from the second set. The kernel K_{ij} is a suitable loosely coherent kernel. In the present implementation we have explored exponential kernel described in [5], the sinc kernel and two variations of the Lanczos kernels. The latter have been chosen for regular use as it provides the best tradeoff between flatness for a limited amount phase mismatch and keeping the most coefficients of K_{ij} zero.

The kernel presently used is based on Lanczos kernel with parameter 3

defined in function `lanczos_kernel3`:

$$\tilde{K}_{ij}(\delta) = \begin{cases} \delta\,|t_i - t_j|\,/1800.0 \text{ sec} > 3.0 & 0.0 \\ \text{else} & \text{sinc}(\delta\,|t_i - t_j|\,/1800.0 \text{ sec})\text{sinc}(\delta\,|t_i - t_j|\,/5400.0 \text{ sec}) \end{cases}$$

where $\text{sinc}(x) = \frac{sin(x)}{x}$ is the well-known *sinc* function. t_i are the GPS times of the SFTs being summed and δ is the parameter describing tolerance to the phase mismatch.

The full kernel takes into account weights and amplitude modulation:

$$K_{ij} = F\sqrt{w_j}\tilde{K}_{ji}\sqrt{w_i}$$

where w_i are the conventional inverse square PowerFlux weights and A is one of quadratic amplitude modulation factors F_+^2, $F_\times^2$ or $F_+F_\times$. The imaginary component of amplitude modulation is neglected to fit the existing PowerFlux array layout (more details are in section 6.

Neither background subtraction nor line avoidance are implemented.

5.2 Caching of partial power sums

The cached accumulation function follows the same algorithm as the one for `single_bin` power sum method, except that key depends on differential shift as well as a regular one - this is to account for phase variance between neighbouring frequency bins.

5.3 Power sum accumulation

The power sum accumulation function is different in two respects: first, it is a double sum over SFT sets from each group. Secondly, the block is partitioned into groups both by sidereal time (as in `single_bin` case) and, in addition, by time as well. The latter is done to speed up computation - for large δ values there is no reason to compute matrix elements between groups widely spaced SFT sets.

Also, the default value of `summing_step` parameter that describes the length of one block is 3 days, not 10 as for single bin code.

6 Processing of amplitude modulation in PowerFlux2

There are two ways to derive the formulas for universal polarization coefficients appropriate for computing bilinear products. One way is to apply mathematical technique of polarization of homogeneous polynomials, while the other is to derive it from the basics.

We will follow the formalism of [1, 6, 7, 8]

Let us consider a single SFT and an elliptically polarized source with major axis tilted at an angle ψ w.r.t. vertical axis.

We assume that during SFT period (usually 30 minutes or less) the frequency of source can be assumed constant.

$$\begin{aligned} h'_+ &= A_+ \cos(\omega t) \\ h'_\times &= A_\times \sin(\omega t) \end{aligned}$$

A generic pulsar signal can be represented as $A_+ = h_0 \left(1 + \cos^2(\iota)\right)/2$, $A_\times = h_0 \cos(\iota)$, with $h_0 = A_+ + \sqrt{A_+^2 - A_\times^2}$ and $\cos(\iota) = A_\times / \left(A_+ + \sqrt{A_\times^2 - A_+^2}\right)$

We will assume that demodulation is performed for a fixed frame of plus and cross polarizations rotated at an angle α. In this coordinate system we have:

$$\begin{aligned} h_+ &= A_+ \cos(\omega t)\cos(\epsilon) - A_\times \sin(\omega t)\sin(\epsilon) \\ h_\times &= A_+ \cos(\omega t)\sin(\epsilon) + A_\times \sin(\omega t)\cos(\epsilon) \end{aligned}$$

where we introduced $\epsilon = 2(\psi - \alpha)$.

The signal amplitude in SFT bin corresponding to frequency ω is then

$$\begin{aligned} z &= \int \left(F_+ h_+ + F_\times h_\times\right) e^{-i\omega t} dt = \\ &= \tfrac{1}{2}\left(F_+(A_+\cos(\epsilon) + iA_\times \sin(\epsilon)) + F_\times(A_+ \sin(\epsilon) - iA_\times \cos(\epsilon))\right) \\ &= \tfrac{1}{2}\left(A_+(F_+\cos(\epsilon) + F_\times \sin(\epsilon)) + iA_\times(F_+ \sin(\epsilon)) - F_\times \cos(\epsilon))\right) \end{aligned}$$

For the computation of loosely coherent statistic we are concerned with the product $z^1 \bar{z}^2$ of signal amplitudes from two SFTs.

$$\begin{aligned} \mathrm{Re} z^1 \bar{z}^2 &= \tfrac{1}{4}\mathrm{Re}\left(A_+(F_+^1\cos(\epsilon) + F_\times^1 \sin(\epsilon)) + iA_\times(F_+^1 \sin(\epsilon) - F_\times^1 \cos(\epsilon))\right) \cdot \\ &\quad \cdot \left(A_+(F_+^2\cos(\epsilon) + F_\times^2 \sin(\epsilon)) - iA_\times(F_+^2 \sin(\epsilon) - F_\times^2 \cos(\epsilon))\right) = \\ &= \tfrac{1}{4}\left(A_+^2(F_+^1\cos(\epsilon) + F_\times^1 \sin(\epsilon))(F_+^2\cos(\epsilon) + F_\times^2 \sin(\epsilon)) + \right. \\ &\quad \left. + A_\times^2(F_+^1 \sin(\epsilon)) - F_\times^1 \cos(\epsilon))(F_+^2 \sin(\epsilon)) - F_\times^2 \cos(\epsilon))\right) \end{aligned}$$

Grouping terms with products of F_+ and $F_\times$ we obtain:

$$\begin{aligned}
\mathrm{Re}z^1\bar{z}^2 &= \tfrac{1}{4}\left(A_+^2(F_+^1\cos(\epsilon)+F_\times^1\sin(\epsilon))(F_+^2\cos(\epsilon)+F_\times^2\sin(\epsilon))+\right.\\
&\quad \left.+A_\times^2(F_+^1\sin(\epsilon))-F_\times^1\cos(\epsilon))(F_+^2\sin(\epsilon)-F_\times^2\cos(\epsilon))\right)=\\
&= \tfrac{1}{4}\left(F_+^1F_+^2(A_+^2\cos^2(\epsilon)+A_\times^2\sin^2(\epsilon))+F_+^1F_\times^2(A_+^2-A_\times^2)\cos(\epsilon)\sin(\epsilon)+\right.\\
&\quad \left.+F_\times^1F_+^2(A_+^2-A_\times^2)\cos(\epsilon)\sin(\epsilon)+F_\times^2F_\times^2(A_+^2\sin^2(\epsilon)+A_\times^2\cos^2(\epsilon))\right)=\\
&= \tfrac{1}{4}\left(F_+^1F_+^2(A_+^2\cos^2(\epsilon)+A_\times^2\sin^2(\epsilon))+\right.\\
&\quad +\tfrac{1}{2}(F_+^1F_\times^2+F_\times^1F_+^2)(A_+^2-A_\times^2)\sin(2\epsilon)+\\
&\quad \left.+F_\times^1F_\times^2(A_+^2\sin^2(\epsilon)+A_\times^2\cos^2(\epsilon))\right)=\\
&= \tfrac{1}{8}\left((F_+^1F_+^2+F_\times^1F_\times^2)(A_+^2+A_\times^2)+\right.\\
&\quad +(F_+^1F_+^2-F_\times^1F_\times^2)(A_+^2-A_\times^2)\cos(2\epsilon)+\\
&\quad \left.+(F_+^1F_\times^2+F_\times^1F_+^2)(A_+^2-A_\times^2)\sin(2\epsilon)\right)
\end{aligned}$$

Similarly

$$\begin{aligned}
\mathrm{Im}z^1\bar{z}^2 &= \tfrac{1}{4}\mathrm{Im}\left(A_+(F_+^1\cos(\epsilon)+F_\times^1\sin(\epsilon))+iA_\times(F_+^1\sin(\epsilon)-F_\times^1\cos(\epsilon))\right)\cdot\\
&\quad \cdot\left(A_+(F_+^2\cos(\epsilon)+F_\times^2\sin(\epsilon))-iA_\times(F_+^2\sin(\epsilon)-F_\times^2\cos(\epsilon))\right)=\\
&= \tfrac{1}{4}\left(F_+^1F_\times^2A_+A_\times-F_\times^1F_+^2A_+A_\times\right)=\\
&= \tfrac{1}{4}\left(F_+^1F_\times^2-F_\times^1F_+^2\right)A_+A_\times
\end{aligned}$$

Thus the response of a single SFT bin can be decomposed into detector response terms $F_+^1F_+^2+F_\times^1F_\times^2$, $F_+^1F_+^2-F_\times^1F_\times^2$, $F_+^1F_\times^2+F_\times^1F_+^2$ and $F_+^1F_\times^2-F_\times^1F_+^2$ and corresponding polarization dependent coefficients that are universal for all SFTs. If one computes partial power sums for detector response terms alone - an effort equivalent to sampling 3-5 individual polarizations - then it is possible to reconstruct power sums for any polarization without summing over the entire SFT set.

The single bin code is only concerned with the case $z_1 = z_2$ in which case the imaginary component is identically zero.

The imaginary component is neglected in loosely coherent search code as original partial power sum structures were constructed for single bin case alone. This does not result in a large impact for two reasons:

- The antenna pattern factors evolve slowly in time making it small for closely spaced SFTs. For $\pi/2$ search the kernel coefficients is less than 5% of maximum value for SFTs 2 hours apart. For $\pi/5$ search the distance is 5 hours.

- The product $A_+A_\times$ is small for linear polarizations which make up the overall upper limit. Thus the losses from not taking $\mathrm{Im}z^1\bar{z}^2$ into

account for elliptically polarized signals are made up by overestimate when demodulating them as linearly polarized signals.

The overall correction is a combination of these two factors. For example, when considering $\pi/2$ search we can bound the effect of evolution of amplitude modulation factors as 2 hours/24 hours = 8.3%. As the power injected by the signal varies by a factor of 8 between linearly polarized signals and circularly polarized signals the overall upper limit will not be affected.

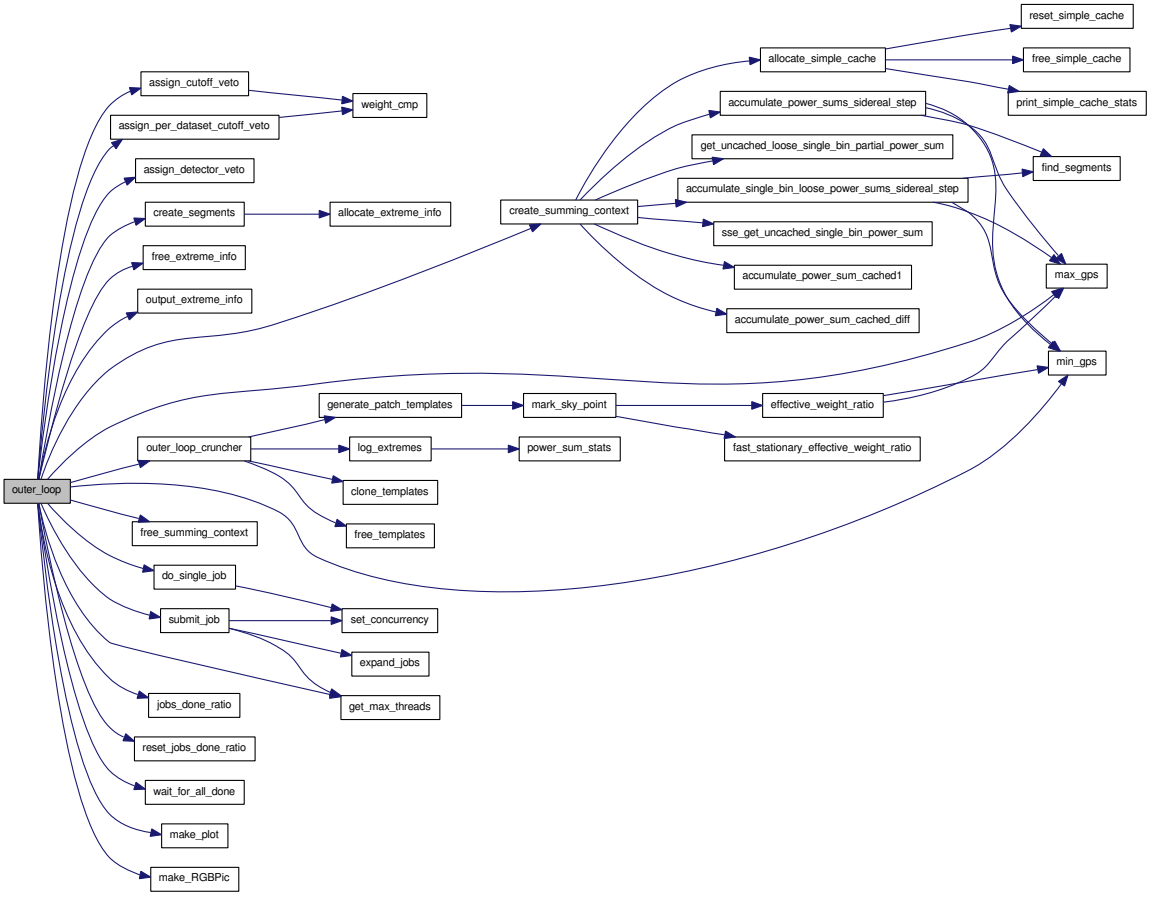

Figure 11: Call graph of PowerFlux2 outer loop function. This does not include calls to methods of summing context.

7 Example output

The figures 12 and 14 contain example output produced using configuration files shown on figures 15, 16 and 17.

In all cases we have made a linearly polarized injection into Gaussian noise that spanned several months. The injection strength was chosen to produce a signal-to-noise ratio of 8.29 - a typical value for an outlier to be followed up. Figure 12 shows the entire sky as would be explored in typical PowerFlux run. Figure 14 shows a skymap produced using regular single-bin code (left) and a skymap made with loosely coherent search with $\delta = \pi/2$ and 10x magnification.

Figure 13 shows corresponding skymap of upper limits. The injection is barely visible in the top right corner due to less sensitive region at the equator.

On all skymaps the square pixels describe patches - multiple templates were sampled for each patch.

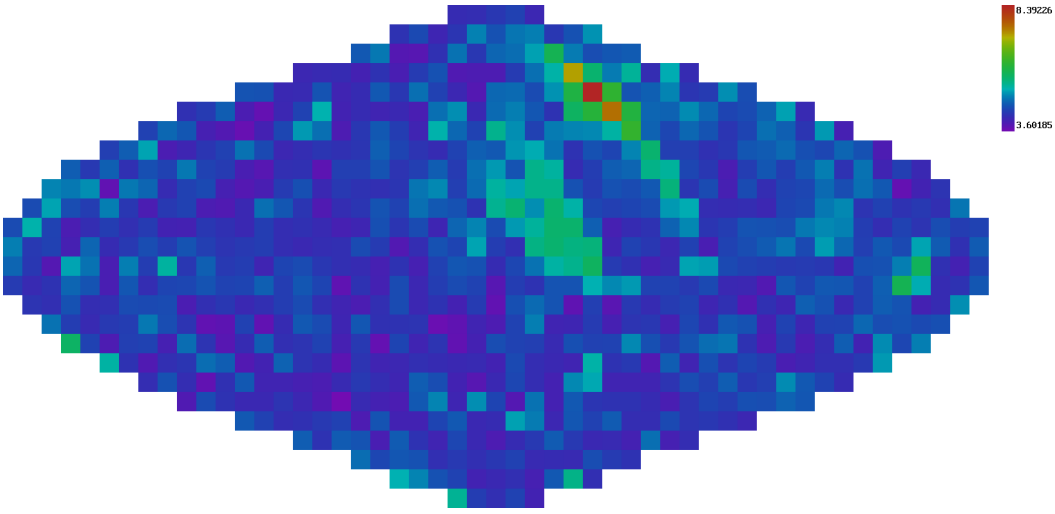

Figure 12: Example skymap of signal-to-noise ratios using single-bin PowerFlux2 mode. The injection was performed at RA=2.0 and DEC=1.0. Each square corresponds to a sky patch - a set of templates that used the same amplitude response coefficients.

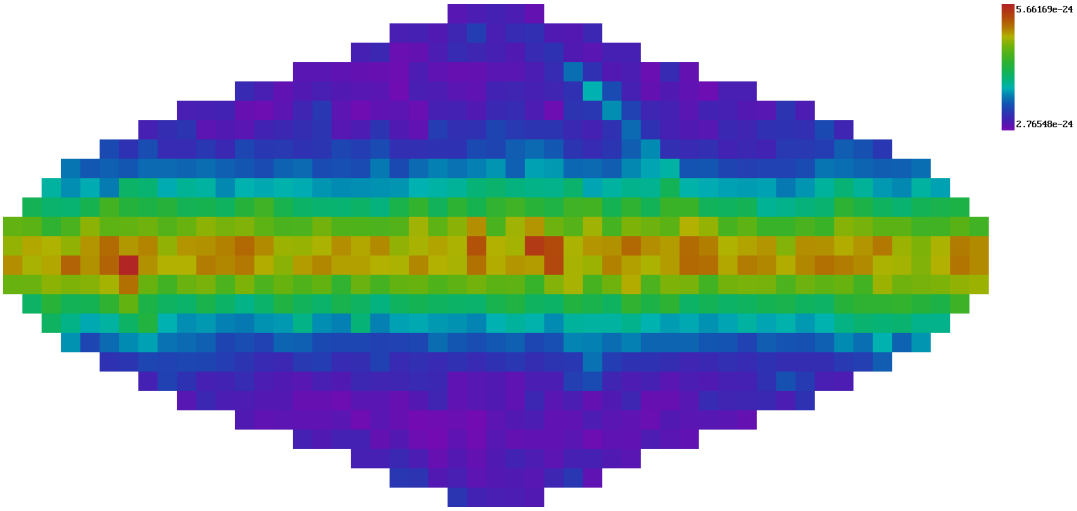

Figure 13: Example skymap of upper limits using single-bin PowerFlux2 mode. The injection was performed at RA=2.0 and DEC=1.0 and is barely visible, compared to equatorial points where we are less sensitive.

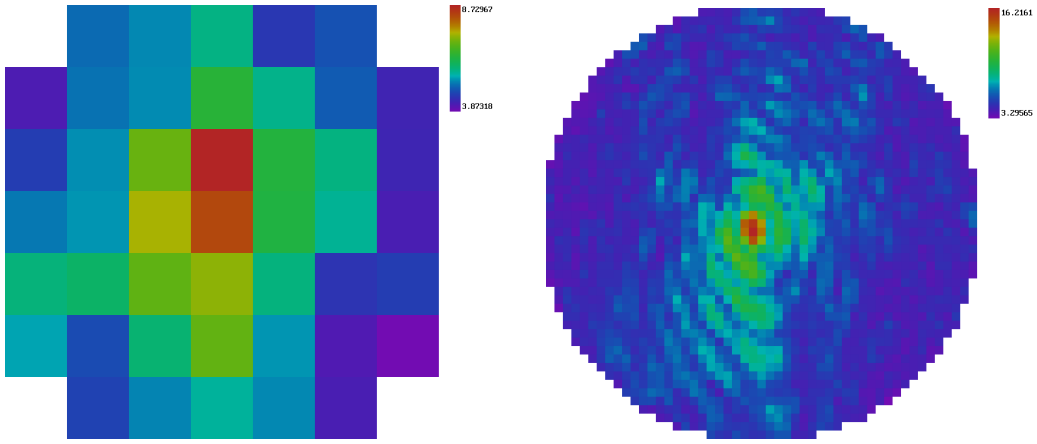

Figure 14: Zoomed of the injection. The area displayed in both pictures is a disk with 0.3 radians radius centered at the location of the outlier of figure 12. The left image has been made with single-bin code. The right image was produced by loosely coherent code using $\delta = \pi/2$ and `sky-resolution-ratio`=0.1.

8 PowerFlux configuration options

When started with `--help` option PowerFlux prints out a summary of available command line switches:

```
powerflux 1.4.39-64

Powerflux analysis program

Usage: powerflux [OPTIONS]...

  -h, --help                    Print help and exit
  -V, --version                 Print version and exit
  -c, --config=STRING           configuration file (in gengetopt format) to
                                  pass parameters
      --label=STRING            arbitrary string to be printed in the beginning
                                  of PowerFlux log file  (default=‘’)
      --sky-grid=STRING         sky grid type (arcsin, plain_rectangular,
                                  sin_theta)  (default=‘sin_theta’)
      --skymap-orientation=STRING
                                orientation of produced skymaps: equatorial,
                                  ecliptic, band_axis  (default=‘equatorial’)
      --skyband-method=STRING   method of assigning band numbers: angle, S
                                  (default=‘S’)
      --nskybands=INT           split sky in this many bands for logging
                                  maximum upper limits  (default=‘11’)
      --large-S=DOUBLE          value of S to consider good enough
      --band-axis=STRING        which band axis to use for splitting sky into
                                  bands (perpendicular to band axis) (possible
                                  values: equatorial, auto,
                                  explicit(float,float,float)  (default=‘auto’)
      --band-axis-norm=DOUBLE   norm of band axis vector to use in S value
                                  calculation
      --sky-marks-file=STRING   file describing how to mark up a sky
      --fine-factor=INT         make fine grid this times finer  (default=‘5’)
      --skymap-resolution=DOUBLE
                                specify skymap resolution explicitly
      --skymap-resolution-ratio=DOUBLE
                                adjust default coarseness of the grid by this
                                  factor  (default=‘1.0’)
      --small-weight-ratio=DOUBLE
```

```
                              ratio that determines which weight is too small
                                to include in max statistics  (default=‘0.2’)
    --strain-norm-factor=DOUBLE
                              strain normalization factor to prevent
                                overflowing of the exponent
                                (default=‘1e-20’)
    --lock-file=STRING        file to lock when reading SFTs in order to
                                globally serialize disk access
    --enable-dataset-locking=INT
                              set to 1 to enable dataset level locking
                                (default=‘1’)
    --retry-delay=INT         number of seconds to wait before retrying I/O
                                (default=‘2’)
    --lock-retry-delay=INT    number of seconds to wait before trying to
                                acquire lock again  (default=‘10’)
-s, --dataset=STRING          dataset file
-i, --initial-dataset-seed=INT
                              initial seed to use for generating gaussian
                                data  (default=‘12345’)
    --input-format=STRING     format of input files (GEO, SFT, Power)
                                (default=‘GEO’)
    --dump-data=STRING        file to output loaded SFT data into, for
                                testing
    --dump-sftv2=STRING       directory to output loaded data, together with
                                dataset description
-o, --output=STRING           output directory
    --ephemeris-path=STRING   path to detresponse program from lalapps
    --earth-ephemeris=STRING  Earth ephemeris file, overrides ephemeris-path
                                argument
    --sun-ephemeris=STRING    Sun ephemeris file, overrides ephemeris-path
                                argument
-f, --first-bin=INT           first frequency bin in the band to be analyzed
-n, --nbins=INT               number of frequency bins to analyze
                                (default=‘501’)
    --side-cut=INT            number of bins to cut from each side due to
                                corruption from doppler shifts
    --expected-timebase=DOUBLE
                              expected timebase in months  (default=‘6’)
    --hist-bins=INT           number of bins to use when producing histograms
                                 (default=‘200’)
```

```
-d, --detector=STRING          detector location (i.e. LHO or LLO), passed to
                                 detresponse
    --doppler-multiplier=DOUBLE
                               a constant to multiply Doppler shifts by (1.0
                                 corresponds to standard physics)
                                 (default=‘1.0’)
    --spindown-start-time=DOUBLE
                               specify spindown start time in GPS sec. Assumed
                                 to be the first SFT segment by default
    --frequency-offset=DOUBLE (small) frequency offset - used to achieve
                                 fractional bin shifts  (default=‘0.0’)
    --spindown-start=DOUBLE    first spindown value to process
                                 (default=‘0.0’)
    --spindown-step=DOUBLE     step for processing multiple spindown values
                                 (default=‘5e-10’)
    --spindown-count=INT       how many separate spindown values to process
                                 (default=‘1’)
    --fdotdot=DOUBLE           second frequency derivative  (default=‘0.0’)
    --orientation=DOUBLE       additional orientation phase, specifying 0.7853
                                 will turn plus into cross  (default=‘0’)
    --nlinear-polarizations=INT
                               even number of linear polarizations to profile,
                                 distributed uniformly between 0 and PI/2
                                 (default=‘4’)
    --no-demodulation=INT      do not perform demodulation stage, analyze
                                 background only  (default=‘0’)
    --no-decomposition=INT     do not perform noise decomposition stage,
                                 output simple statistics only  (default=‘0’)
    --no-candidates=INT        do not perform analysis to identify candidates
                                 (default=‘0’)
    --no-am-response=INT       force AM_response() function to return 1.0
                                 irrespective of the arguments  (default=‘0’)
    --no-secondary-skymaps=INT
                               do not store values not essential for upper
                                 limits and followup  (default=‘0’)
    --averaging-mode=STRING    1 - use one bin, 3 - average 3, matched - use 7
                                 bin matched filter  (default=‘1’)
    --subtract-background=INT subtract rank 1 matrix in order to flatten
                                 noise spectrum  (default=‘0’)
    --do-cutoff=INT            neglect contribution from SFT with high
```

```
                                effective noise level  (default='1')
      --filter-lines=INT        perform detection of lines in background noise
                                and veto corresponding frequency bins
                                (default='1')
      --ks-test=INT             perform Kolmogorov-Smirnov test for normality
                                of averaged powers  (default='1')
      --compute-betas=INT       compute beta coefficients as described in
                                PowerFlux polarizations document
                                (default='0')
      --upper-limit-comp=STRING upper limit compensation factor - used to
                                account for windowing in SFTs (possible
                                values: Hann, flat, arbitrary number)
                                (default='Hann')
      --lower-limit-comp=STRING lower limit compensation factor - used to
                                account for windowing in SFTs (possible
                                values: Hann, flat, arbitrary number)
                                (default='Hann')
      --write-dat=STRING        regular expression describing which *.dat files
                                to write  (default='.*')
      --write-png=STRING        regular expression describing which *.png files
                                to write  (default='.*')
      --dump-points=INT         output averaged power bins for each point in
                                the sky  (default='0')
      --dump-candidates=INT     output SFT data for first N candidates
                                (default='0')
      --focus-ra=DOUBLE         focus computation on a circular area with
                                center at this RA
      --focus-dec=DOUBLE        focus computation on a circular area with
                                center at this DEC
      --focus-radius=DOUBLE     focus computation on a circular area with this
                                radius
      --only-large-cos=DOUBLE   restrict computation to points on the sky with
                                cos of angle to band axis larger than a given
                                number

 Group: injection
      --fake-linear             Inject linearly polarized fake signal
      --fake-circular           Inject circularly polarized fake signal
      --fake-ref-time=DOUBLE    time of signal start  (default='0')
      --fake-ra=DOUBLE          RA of fake signal to inject  (default='3.14')
```

```
--fake-dec=DOUBLE          DEC of fake signal to inject  (default=‘0.0’)
--fake-iota=DOUBLE         iota of fake signal to inject  (default=‘0.0’)
--fake-psi=DOUBLE          orientation of fake signal to inject
                             (default=‘0.0’)
--fake-phi=DOUBLE          phase of fake signal to inject  (default=‘0.0’)
--fake-spindown=DOUBLE     spindown of fake signal to inject
                             (default=‘0.0’)
--fake-strain=DOUBLE       amplitude of fake signal to inject
                             (default=‘1e-23’)
--fake-freq=DOUBLE         frequency of fake signal to inject
--snr-precision=DOUBLE     Assumed level of error in detection strength -
                             used for listing candidates  (default=‘0.2’)
--max-candidates=INT       Do not optimize more than this number of
                             candidates  (default=‘-1’)
--min-candidate-snr=DOUBLE
                           Do not optimize candidates with SNR below this
                             level  (default=‘5.0’)
--output-initial=INT       write initial candidates into log file
                             (default=‘0’)
--output-optimized=INT     write optimized (second pass) candidates into
                             log file  (default=‘0’)
--output-cache=INT         write out all candidates in cache to log file
                             (default=‘0’)
--extended-test=INT        Perform extended self test  (default=‘0’)
--max-sft-report=INT       Maximum count of SFTs to report with veto
                             information  (default=‘100’)
--num-threads=INT          Use that many threads for computation
                             (default=‘-1’)
--niota=INT                Number of iota values to use in alignment grid
                             (default=‘3’)
--npsi=INT                 Number of psi values to use in alignment grid
                             (default=‘6’)
--nfshift=INT              Number of sub-bin frequency shifts to sample
                             (default=‘2’)
--nchunks=INT              Partition the timebase into this many chunks
                             for sub period analysis  (default=‘5’)
--split-ifos=INT           Split interferometers in separate chunks
                             (default=‘1’)
--weight-cutoff-fraction=DOUBLE
                           Discard sfts with small weights that contribute
```

```
                              this fraction of total weight
                              (default=‘0.04’)
--per-dataset-weight-cutoff-fraction=DOUBLE
                            Discard sfts with small weights that contribute
                              this fraction of total weight in each dataset
                               (default=‘0.04’)
--power-max-median-factor=DOUBLE
                            This determines scaling factor between median
                              and maximum of exponentially distributed
                              variable. Used for computing power sum
                              weights  (default=‘0.1’)
--tmedian-noise-level=INT Use TMedians to estimate noise level (as
                              opposed to in-place standard deviation)
                              (default=‘1’)
--summing-step=DOUBLE     integration step size, in seconds
--max-first-shift=INT     larger values accomodate bigger spindown ranges
                              but require more bins to be computed in
                              uncached function  (default=‘10’)
--statistics-function=STRING
                            specify statistics postprocessing to apply.
                              Possible values: linear, sorted
                              (default=‘linear’)
--dump-power-sums=INT     Write out all power sum data into data.log
                              file. It is recommend to restrict the sky to
                              very few pixels  (default=‘0’)
--compute-skymaps=INT     allocate memory and compute skymaps with final
                              results  (default=‘0’)
--fine-grid-skymarks=INT  use sky marks from the fine grid, this uses
                              constant spindown  (default=‘0’)
--half-window=INT         number of bins to exclude to the left and to
                              the right of highest point when computing
                              linear statistics  (default=‘20’)
--tail-veto=INT           do not report outlier if its frequency is
                              within that many bins from the tail - happens
                              with steep spectrum  (default=‘10’)
--cache-granularity=INT   granularity of power cache frequency shift
                              resolution, in fractions of a frequency bin
                              (default=‘-1’)
--sidereal-group-count=INT
                            separate SFTs in that many groups by frequency
```

```
                              shift
--time-group-count=INT    separate SFTs in that many groups by gps time
--phase-mismatch=DOUBLE   maximal phase mismatch over coherence length to
                              assume when using loosely coherent mode
                              (default='1.570796')
--bypass-powersum-cache=INT
                            bypass partial power sum cache  (default='0')
```

8.1 config

Specify the first part of SFT file name. The complete name is formed by appending an SFT number to the end.

Instead of specifying arguments on the command line it is possible to create a file with each line containing `option value` pair. The "–" prefix in front of the option name must to be omitted.

In the following PowerFlux options would be referred by their name as used in the configuration file, for command line use prepend `"--"`.

8.2 Input/Output options

8.2.1 config

Configuration file with more options. This can be supplied more than once.

8.2.2 dataset

Specify file containing description of the data to load.

8.2.3 input-format

Specify the format of input SFT files.

- `Power` refers to ASCII header binary body power-only files produces by `make_sft_op`
- `SFT` refers to ASCII header binary body SFT files produced by `make_sft_op`
- `GEO` refers to binary header binary body GEO-style SFT files in common use in LSC.

8.2.4 `segments-file`

Allows to restrict processing to only SFT files that are inside a list of segments described in the ASCII file specified with this option.

Each line specifies a single segment described the starting GPS time followed by ending GPS time. The rest of line is discarded (this makes possible to use standard segment list files).

8.2.5 `veto-segments-file`

Same as `segments-file` except this option specifies the list of times **not** to process. Useful for vetoing parts of data.

8.2.6 `ephemeris-path`

Path to files with ephemeris data (they can be found, for example, in `lalapps/src/detresponse/` directory).

8.2.7 `earth-ephemeris`

Specify Earth ephemeris file explicitly. This overrides `ephemeris-path` option.

8.2.8 `sun-ephemeris`

Specify Sun ephemeris file explicitly. This overrides `ephemeris-path` option.

8.2.9 `skymarks`

Specify file describing sky partitioning into areas for data collection.

8.2.10 `output`

Specify directory to put output of PowerFlux into. Everything that PowerFlux writes will be located in this directory.

8.3 Analysis parameters

8.3.1 `first-bin`

Specify first bin of 501 bin stretch to analyze. This is an integer in units of 1/1800 Hz. It specifies the *source* (i.e. decoded) frequency, not frequency as received by the detector.

8.3.2 `nbins`

Number of frequency bins to analyze. The default is 501. At the moment this value should not be changed - there are some hard-coded constants that rely on this number. In particular, the Feldman-Cousins method relies on constants produced by Monte-Carlo simulation on the assumption that `nbins`= 501.

8.3.3 `side-cut`

Due to the need to apply Doppler shift the actual number of bins read from SFT file is larger than `nbins` and varies with frequency and spindown. Normally PowerFlux will compute the number of extra bins to read automatically. This option allows an explicit override. Specifying `--side-cut=100` will cause PowerFlux to read all bins from `first-bin`−100 to `first-bin`+`nbins`+100.

8.3.4 `spindown-start`

Specify initial spindown value to process. It is a floating-point value in units of Hz/sec. Negative values correspond to frequency decreasing with time.

8.3.5 `spindown-count`

Specify the number of spindown values to process starting with value specified by `spindown-start` option.

8.3.6 `spindown-step`

Specify the increment between spindown values to process. Can be positive or negative.

8.3.7 `nfshift`

Number of sub-bin steps to sample. The frequency resolution will in units of 1/nfshift of frequency bin.

8.3.8 `nchunks`

Break up the loaded dataset into `nchunks` equally spaced segments and report results for any contiguous combination.

8.3.9 `niota`

Sample niota + 1 values, including 0 (for circular polarization) and $\pi/2$ (for linear polarization).

8.3.10 `npsi`

Sample `npsi` orientation values. The total number of polarizations sampled is npsi $\cdot$ niota + 1, as circular polarization does not depend on ψ.

8.3.11 `phase-mismatch`

Maximum allowable phase mismatch for loosely coherent searches.

8.4 Analysis options

8.4.1 `no-demodulation`

Do not perform demodulation, stop after analyzing background. Since Feldman-Cousins is not performed `nbins` can be specified to an arbitrary number, although values exceeding 25 Hz require lots of computer memory (in excess of 2 Gb).

It is convenient to specify `--side-cut=0` to provide greater control over starting frequency.

8.4.2 `no-decomposition`

Only read in SFT files and output simple statistics. This is even faster than `no-demodulation` option. Same suggestions apply.

8.4.3 `no-am-response`

Assume that amplitude response is always 1.0 irrespective of time or sky position.

8.4.4 `skymap-resolution`

PowerFlux computes optimal resolution of skymaps automatically (It depends mostly on magnitude of Doppler shifts). This options allows to specify resolution explicitly skymaps. This option is very handy for comparing PowerFlux skymap output for different frequency bands.

8.4.5 `skymap-resolution-ratio`

PowerFlux computes optimal resolution of skymaps automatically (It depends mostly on magnitude of Doppler shifts). This options allows to force skymaps to have a finer or coarser resolution by a given factor.

8.4.6 `small-weight-ratio`

PowerFlux discards data in weighted sum that is assigned too small a weight. This option allows to specify it.

For a perfect instrument setting this ratio to 0 will produce the best result as the weighting scheme used will make good use even of data with very small weights.

In practice, discarding SFTs saves CPU cycles so it makes sense to skip those which provide marginal improvement.

Furthermore, the SFTs with small weight can often have radically different noise spectrum. The default value is prudent 0.2.

8.4.7 `three-bins`

Specifying `three-bins=1` causes PowerFlux to average every neighbouring three bins in its analysis. Because of this Doppler tracks are widened and a coarser skymaps may be used while still retaining full sky coverage.

The drawback is a factor of $\sqrt{2}$ loss in sensitivity.

8.4.8 `do-cutoff`

Enabled by default. Setting it to 0 will turn off Cutoff computation. This is similar to specifying `--small-weight-ratio=0.0`

8.4.9 `filter-lines`

Perform automatic detection of lines in background noise and veto corresponding frequency bins. Up to 5 frequency bins can be vetoed.

8.4.10 `subtract-background`

Subtract rank 1 matrix formed from `TMedians` and `FMedians` in order to improve noise performance for sky positions with large variation in Doppler shifts. Turning this option on will clobber signals from sky positions with small variation in Doppler shifts, therefore one should combine it with `only-large-cos=0.3`.

8.5 Data reporting options

8.5.1 `skymap-orientation`

The skymaps produced by PowerFlux can have different orientations to please the user. Possible choices are equatorial, ecliptic or "band-axis" - with band axis vector pointing to the North pole.

8.5.2 `nskybands`

Split sky in a given number of bands and report analysis results for each band individually.

8.5.3 `skyband-method`

Specify method used to partition sky into regions.

Possible values are `angle` and `S`. The latter method is useful on short timebases, while the former can used to partition the sky into bands along declination.

8.5.4 `band-axis`

By default PowerFlux computes optimal band axis automatically (this has to do with average detector acceleration during analyzed data set). However, it may be useful to specify it explicitly - for example for comparison of results between different interferometers.

Possible values are `equatorial`, `auto` and `explicit(%f,%f,%f)`.

8.5.5 `band-axis-norm`

Specify the norm of band axis vector explicitly. This is useful for comparison of results between different IFOs.

8.5.6 `large-S`

Specify values of S function considered to be good enough. All sky points with S value larger or equal to this value will be assigned to band 0.

8.5.7 `only-large-cos`

Restrict computation to only those areas of sky which projection to band axis has absolute value larger than a value specified to this option. If you want to do this (due to presence of line artifacts, for example) the recommended value is 0.3.

This cleans up the results reported for entire skymap and can significantly reduce computation time requirements.

8.5.8 `ks-test`

Perform and output results of Kolmogorov-Smirnov test for compliance of averaged weighted power with gaussian distribution with parameters employed later to establish Feldman-Cousins limits. This increases the computation time, but is a highly recommended cross check for analysis. High values of KS statistic indicate bands with pathological noise floor behaviour.

8.5.9 `upper-limit-comp`

A factor to multiply upper limits reported in strain units. One can specify a floating-point number or "Hann" for Hann windowed SFTs. This factor is used to account for the fact that non bin-centered (in frequency) signals

would have smaller amplitude than bin-centered ones. For Hann windowed SFTs, using 1-bin mode the factor is 1/0.85.

8.5.10 `lower-limit-comp`

A factor to multiply lower limits reported in strain units. One can specify a floating-point number or "Hann" for Hann windowed SFTs. This factor is used to account for the fact that non bin-centered (in frequency) signals would have smaller amplitude than bin-centered ones. For Hann windowed SFTs, using 1-bin mode the factor is 1.

8.5.11 write-dat

By default PowerFlux writes a binary file with data for each plot it makes. You can use this option to specify a regular expression to filter what will actually be written.

This can significantly reduce storage requirements for PowerFlux output, as well as speed up computation.

8.5.12 write-png

By default PowerFlux creates a number of plots. You can use this option to specify a regular expression to filter what will actually be written.

This can significantly reduce storage requirements for PowerFlux output, as well as speed up computation.

8.6 Software injections

The following options provide interface to software injections done by PowerFlux itself (as opposed to using external programs). The injections are power-only, modeled with assumption of random phase of incoming signal to a particular frequency bin.

8.6.1 `fake-linear`

Perform injection of linearly polarized signal.

8.6.2 `fake-circular`

Perform injection of circularly polarized signal.

8.6.3 `fake-ra`

Specify right ascension of injected signal source in radians (values from 0 to 2π are acceptable).

8.6.4 `fake-dec`

Specify declination of injected signal source in radians (values from $-\pi/2$ to $\pi/2$ are acceptable).

8.6.5 `fake-orientation`

Specify polarization of injected signal (assumed to be linearly polarized). Valid values are between 0 and $\pi/4$.

8.6.6 `fake-spindown`

Specify spindown of injected signal in units of Hz/sec.

8.6.7 `fake-strain`

Specify strain of injected signal.

8.6.8 `fake-freq`

Specify frequency of injected signal.

References

[1] V. Dergachev, "Description of PowerFlux Algorithms and Implementation", LIGO technical document LIGO-T050186 (2005), available in `http://admdbsrv.ligo.caltech.edu/dcc/`

[2] All-sky search for periodic gravitational waves in LIGO S4 data, B. Abbott *et al.* (The LIGO Scientific Collaboration), Phys. Rev. D **77**, 022001 (2008).

[3] All-sky LIGO Search for Periodic Gravitational Waves in the Early S5 Data, B. Abbott *et al.* (The LIGO Scientific Collaboration), Phys. Rev. Lett. 102, 111102 (2009)

```
dataset random.dst
input-format GEO
earth-ephemeris earth05-09.dat
sun-ephemeris sun05-09.dat
sky-marks all_sky_marks.txt
first-bin 180000
nbins 501
do-cutoff 1
filter-lines 1
averaging-mode one
spindown-start-time 793154935
spindown-start -1e-8
expected-timebase 7

write-dat NONE
#write-png NONE
subtract-background 1
ks-test 1
compute-betas 0
output-initial 1
max-candidates 0

fake-ref-time 793154935
fake-ra 2.0
fake-dec 1.0
fake-iota 1.570796
fake-psi 0.392699
fake-phi 0.0
fake-spindown -1e-8
fake-strain 3e-24
fake-freq 100.1389

skymap-resolution-ratio 1

compute-skymaps 1

initial-dataset-seed 0

nchunks 1
fine-factor 5

output single_bin
```

Figure 15: `config.single_bin`

```
dataset random.dst
input-format GEO
earth-ephemeris earth05-09.dat
sun-ephemeris sun05-09.dat
sky-marks all_sky_marks.txt
sky-grid targeted_rectangular
first-bin 180000
nbins 501
do-cutoff 1
filter-lines 1
averaging-mode one
spindown-start-time 793154935
spindown-start -1e-8
expected-timebase 7

write-dat NONE
#write-png NONE
subtract-background 1
ks-test 1
compute-betas 0
output-initial 1
max-candidates 0

fake-ref-time 793154935
fake-ra 2.0
fake-dec 1.0
fake-iota 1.570796
fake-psi 0.392699
fake-phi 0.0
fake-spindown -1e-8
fake-strain 3e-24
fake-freq 100.1389

focus-ra 1.978040
focus-dec 1.009798
focus-radius 0.3

skymap-resolution-ratio 1

compute-skymaps 1

initial-dataset-seed 0

nchunks 1
fine-factor 5

output single_bin_zoomed
```

Figure 16: `config.single_bin_zoomed`

```
dataset random.dst
input-format GEO
earth-ephemeris earth05-09.dat
sun-ephemeris sun05-09.dat
sky-marks all_sky_marks.txt
sky-grid targeted_rectangular
first-bin 180000
nbins 501
do-cutoff 1
filter-lines 0
averaging-mode single_bin_loose
spindown-start-time 793154935
spindown-start -1e-8

expected-timebase 7
write-dat NONE
#write-png NONE
subtract-background 1
ks-test 1
compute-betas 0
output-initial 1
max-candidates 0

fake-ref-time 793154935
fake-ra 2.0
fake-dec 1.0
fake-iota 1.570796
fake-psi 0.392699
fake-phi 0.0
fake-spindown -1e-8
fake-strain 3e-24
fake-freq 100.1389

focus-ra 1.978040
focus-dec 1.009798
focus-radius 0.3
skymap-resolution-ratio 0.1

compute-skymaps 1

initial-dataset-seed 0

nchunks 1
fine-factor 5

output loose_pi_2
```

Figure 17: `config.loose_pi_2`

```
#
# Mark general sky areas
#

band "south" "" 0.0 1.570796 -2.0 -0.65
band "midsouth" "" 0.0 1.570796 -0.65 -0.3
band "equator" "" 0.0 1.570796 -0.3 0.3
band "midnorth" "" 0.0 1.570796 0.3 0.65
band "north" "" 0.0 1.570796 0.65 2.0

#
# Using J2000, equatorial coordinates
# All numbers in radians
#

#disk "North_pole" "" 4.712389 1.16 0.05
#disk "South_pole" "" 1.570796 -1.16 0.05
line_response "Lines" "" 0.05 3

# Mask
#band "" "" 1.570796 0.0 -2 0
#band "" "" 2.199115 0.0 0 2
```

Figure 18: `all_sky_marks.txt`

```
new_dataset "H1_test1"
detector "LHO"
gaussian_fill 793154935 900 20001 1e-24
apply_hanning_filter
```

Figure 19: `random.dst`

[4] Unified approach to the classical statistical analysis of small signals, G. J. Feldman and R. D. Cousins, *Phys.Rev.* D **57**, 3873 (1998).

[5] On blind searches for noise dominated signals: a loosely coherent approach, Vladimir Dergachev, arXiv:1003.2178v1 [gr-qc]

[6] Data analysis of gravitational-wave signals from spinning neutron stars. I. The signal and its detection. P. Jaranowski, A. Królak, and B. F. Schutz, Phys. Rev. D **58**, 063001 (1998).

[7] V. Dergachev and K. Riles, "PowerFlux Polarization Analysis" LIGO Technical Document LIGO-T050187 (2005), available in `http://admdbsrv.ligo.caltech.edu/dcc/`

[8] Using generalized PowerFlux methods to estimate the parameters of periodic gravitational waves, Gregory Mendell and Karl Wette, Class. Quantum Grav. 25 (2008) 114044

[9] First upper limits from LIGO on gravitational wave bursts, B. Abbott *et al.* (The LIGO Scientific Collaboration), Phys. Rev. D **69** 102001 (2004).

[10] Limits on Gravitational-Wave Emission from Selected Pulsars Using LIGO Data B. Abbott *et al.* (The LIGO Scientific Collaboration), M. Kramer, and A. G. Lyne, Phys. Rev. Lett. **94** 181103 (2005).

[11] Upper limits on gravitational wave emission from 78 radio pulsars B. Abbott *et al.* (The LIGO Scientific Collaboration), M. Kramer, and A. G. Lyne, Phys Rev. D **76**, 042001 (2007).

[12] Searches for periodic gravitational waves from unknown isolated sources and Scorpius X-1: Results from the second LIGO science run B. Abbott *et al.* (The LIGO Scientific Collaboration), Phys. Rev. D 76 (2007) 082001

[13] The Einstein@Home project is built upon the BOINC (Berkeley Open Infrastructure for Network Computing) architecture described at `http://boinc.berkeley.edu/`.

[14] Einstein@Home search for periodic gravitational waves in LIGO S4 data, B. Abbott *et al.* (The LIGO Scientific Collaboration), Phys. Rev. D 79, 022001 (2009)

[15] Einstein@Home search for periodic gravitational waves in early S5 LIGO data, B. Abbott *et al.* (The LIGO Scientific Collaboration), arXiv:0905.1705

[16] Bernard F.Schutz (1990). A first course in general relativity. Cambridge University Press.

[17] C. W. Misner, K. S. Thorne, and J. A. Wheeler (1973). Gravitation. W. H. Freeman and Co.

[18] Peter R.Saulson (1994). Fundamentals of interferometric gravitational wave detectors. World Scientific.

[19] Making $h(t)$ for LIGO, Xavier Siemens, Bruce Allen , Jolien Creighton , Martin Hewitson, Michael Landry, 2004 Class. Quantum Grav. 21 S1723-S1735