

Data pipeline design and test for the Burst Upper Limits Working Group

AJW's 1st draft 6/1/01, rev. by PRS 6/20/01

Logic of data analysis pipeline

The Burst Group's Upper Limit search will be carried out through the following set of steps:

1. Data from the interferometer is written to frames and also sent to DMT monitors.
2. DMT monitors generate vetoes that establish times when interferometer data should not be searched for signals (ifo out of lock, environment too noisy, dynamic range of key subsystem exceeded,)
3. Frame data enters LDAS pipeline, with the first "scientific" step being Data Conditioning. In the Data Conditioning API, we will:
 - Tag the intervals not to be searched, using information from the DMT
 - Remove spectral lines
 - Whiten the data
 - Accumulate statistics (such as power spectra) to support subsequent analysis.
4. Conditioned data is presented to the bank of burst filters.
5. The outputs of each filter that exceed a certain threshold generate triggers that are written to the database.
6. Triggers are investigated in the Event Analysis Tool. Steps include:
 - Resolution of multiple simultaneous/overlapping triggers into single events
 - Testing for coincidences between multiple LIGO interferometers, other gravitational wave interferometers or bars, and other sources of astrophysical observations (e.g. SNEWS, gamma ray bursts)
 - Investigation of candidate events for untriggered but suspicious signals in the PEM or ifo diagnostic channels
 - Accumulation of event statistics.
7. Calibration into strain units of upper limit or of any detected events.
8. Expressing upper limit or detection in terms of astrophysical models.

(N.B.: This document doesn't cover the analysis track of starting from a list of other astronomical triggers (say, a list of supernova times) and doing the test for excess power in LIGO ifos. That has a whole separate data analysis pipeline that needs to be sketched out.)

Steps necessary to set up and tune the analysis pipeline

For clarity of exposition, we list all of the top-level tuning and testing steps here. In the expanded discussion below, we indicate which of these steps have already been accomplished (or the status of those that haven't.)

1. Frames from engineering runs or other operating times need to be supplied. Also need ability to add fake signals for testing purposes, and routines for generating fake data from scratch (with various statistical properties, with and without fake signals.)

2. Many DMT monitors need to be designed and written.
3. Data conditioning step needs to be designed. Software needs to be tested to ensure that it does no harm.
4. Filtering methods need to be explored and best one(s) chosen. Parameters of filters need to be set.
5. Thresholds need to be set at level that doesn't swamp database but that allows best exploration of signal and noise.
6. Event analysis tool needs to be designed and written. Rules for event resolution need to be derived, and proper coincidence windows determined. Need to established allowable procedures for post-detection study and intervention in event identification or exclusion, or need to establish protocol for tuning procedures in "playpen" data.
7. Dewhitening and calibration procedures need to be written.
8. Establish best ways to link strain output of chosen filters with astrophysical events that can be seen or ruled out.

Work Plan

- Develop a full analysis pipeline for burst searches with multiple LIGO IFOs.
- Test, evaluate and optimize performance of the full analysis pipeline.
- Run full analysis pipeline on E6 data, establish upper limit on coincident event rates (or establish detection of events.)
- Present results in an astrophysically significant way in a publication.

Detailed discussion of work plan, keyed to steps in the data analysis pipeline

Develop a full analysis pipeline for burst searches with multiple LIGO ifos.

1. (Pipeline assumes data is available. Steps needed to generate fake data for testing are described in the section on Testing.)

2. Prepare and test DMT monitors, especially the "flag channel" (saying that data is no good) and the "Data Integrity Monitor" (checking that writing of data is OK, and that monitors of interferometer state are properly functioning.)

3. Prepare data conditioning algorithm, and implement in DataCond API.
 - a. At what stage do we do "gross" vetoes, requiring a locked segment?
 - i. Info on whether a segment is locked is in the DB, and one could query this before the data stretch ever gets into the DataCondAPI; or not pass it from DataCondAPI into WrapperAPI, or let it go all the way through and apply the veto when analyzing the DB triggers with ROOT at the end of the day.
 - b. At what stage do we do "finer" vetoes?
 - i. requiring the IFO to be in lock for some period of time (so that violin resonances damp out).

ii. Using info from other IFO channels correlated with GW channel (e.g., we could look at a seismometer channel that's known to correlate well with the GW channel, and apply a very simple filter in the pipeline.

iii. Using info from DMT filters, via the DB.

c. ADC data has already been pre-whitened somewhat. Do we want to fully whiten (mean=0, SD=1) at this stage? For example, the existing LAL code for excess power requires fully whitened data. We can do that here, or in the WrapperAPI.

d. In order to fully whiten the data, we need the ADC noise curve. Do we accumulate this curve on the fly, e.g., as a running average over the last few minutes, or in a separate run, storing the results in the MetaDB? These curves will be accumulated during data-taking by DMT, but we may want to redo them (for our special needs; to ensure reproducibility; to correct errors or bad assumptions made during data-taking). In any case, the ADC noise curve must be passed to the WrapperAPI.

e. We must also pass the IFO transfer function through the pipeline, so that the WrapperAPI code can work back from excess power to a detected $h(t)$ (if we want it to).

4. Prepare burst filters and implement in LAL/LALWrapper/WrapperAPI. Which classes of filters do we want to implement and evaluate?

a. Classes of filters:

i. Excess power statistic – the only one currently implemented in LAL.

ii. Time-frequency pattern recognition – some of this exists in LAL.

iii. slope filter – very simple and easy to implement.

iv. matched templates – I don't see how this is relevant, unless we have a modeled source that the inspiral group is not doing, like ringdowns.

v. wavelets – can be used as a basis set for excess power. What else?

b. Each filter class must be able to not only generate a S/N and thus a trigger, but also determine parameters: start time; duration; central frequency; bandwidth; absolute signal amplitude (h_{peak} and/or h_{rms}); and confidence level. The calculation of the absolute signal amplitude requires knowledge of the IFO transfer function.

c. Within each class of filters, parameters can vary (eg, $[\Delta t, f_{\text{min}}, \Delta f]$ for excess power statistic). Must build a "bank" of such filters, analogous to inspiral search. The choice of how to "tile" the parameter space of interest is complex, especially since we don't want to rely on a model of what a burst will look like. The "tiling" currently implemented in LAL for the excess power filter seems ad hoc to me, not clear it makes any sense.

d. For each class of filters, we'll have a set of filters in a bank. For these, we need a set of trigger thresholds, to enforce the desired ultimate coincident fake rate. These depend on many things, including:

i. The kind of noise we test the filters on (Gaussian or no).

ii. We can tolerate a relatively high single-IFO fake if we make tighter cuts at the coincidence level (eg, requiring similar waveforms).

iii. We'll want to cut loose on single-IFO triggered events, so that we can vary the single-IFO threshold at the coincidence level, in order to study and optimize it. But we don't want to swamp the DB with countless triggered events with very low S/N!

e. Many choices here: what data segment length and overlap for the LDASjob; what data segment length and overlap for the filters; what algorithm to use for setting the trigger thresholds for each filter in the tiling bank (how to partition the allowed fake rate amongst the many filters in a bank); etc.

f. Learn how to separately tune parameters for, and combine information from, multiple filters within a bank of filters of a given filter class.

5. Prepare form of database entries, and implement in EventMonAPI/MetaDBAPI.

a. Decide whether the database schemas developed by Peter Shawhan and others are already adequate for the task.

i. Segment tables tell us if a segment of data is in lock, etc.

ii. filter and filter params tables define our filters.

iii. sngl_burst and sngl_unmodeled tables are for single-IFO event triggers. Also for bars, SNEWS, and other detectors!

iv. multi_burst table is for multi-detector (IFO's, bars, etc) coincidences.

b. If not, develop new or modified schemas.

c. Each trigger requires unique process_id and filter_id, identifying these entries as being generated by a particular run of LDAS by the burst WG.

d. Prepare, implement and test the mechanisms within LDAS for building database table entries and inserting them in the DB.

6. Prepare algorithm for multi-IFO coincidence analysis, implement (in ROOT?).

a. Extract all relevant records from the database (keying on the table entries that identify this record as relevant to us, eg, because it was generated by a particular run of the burst filter pipeline with unique process_id and filter_id). This includes Segment, Filter, Sngl_burst, Sngl_unmodeled records within the given GPS time interval.

b. Establish live-time: what fraction of real time was each IFO locked, and the data passed through the analysis pipeline, and no coarse vetos applied (violin die-downs, obvious environmental glitches, GDS vetos, etc).

c. Apply any other vetos that may be available at this stage.

d. Construct single-IFO fake rate vs S/N threshold for each filter class, and each filter in the bank for that class.

e. Construct single-IFO signal efficiency vs S/N threshold for each type of signal, for each filter class, and each filter in the bank for that class.

f. Determine, and apply, more stringent S/N threshold, for entry into coincidence analysis. Establish signal efficiencies for those thresholds, for each type signal (eg, ZM supernova as a function of distance).

g. Do the above for two or more IFOs (if real coincident data from two IFOs are not yet available, mock it up with two stretches of data from a single IFO). Inject signals into 2 or more data streams with different time separations.

h. Search for coincidences as a function of coincidence window (time separation).
If found:

i. Deal with multiple-filter triggers of the same event. Choose the filter with the highest S/N? Add them up somehow? Depends on the class of filters...

ii. Check if absolute signal amplitude (h_peak and/or h_rms) are comparable.

- iii. Make any other coincidence cuts deemed appropriate.
- iv. Construct Multi_burst database record and insert into database.
- i. Construct coincidence fake rates (vs some overall S/N threshold), and signal efficiencies (for fixed S/N threshold, vs, eg, distance), for all classes of signals and all classes of filters that have been considered.
- j. Produce examples of final analysis statistics (histograms, upper limits).

7. Calibration into strain units.

8. Connection to astrophysics

- a) We can report fake rates, and signal finding efficiencies, for each burst filter class and each kind of modeled signal.
- b) The trick is to determine what we want on the x-axis. Plotting fake rates as a function of S/N threshold is not astrophysically meaningful.
- c) Some astrophysically-meaningful x-axes:
 - i. h_{\max} or h_{rms} within some specified bandwidth. For example, we can specify one or many $[\Delta t, f_{\min}, \Delta f]$ regions (chosen according to astrophysical bias or more systematic but arbitrary method), and for each, plot event rate vs h_{rms} threshold (which can be turned into upper limits in a straightforward manner), as well as efficiency vs h_{rms} threshold for some representative source models.
 - ii. Instead of specifying $[f_{\min}, \Delta f]$, we can specify "the LIGO bandwidth", and refer to our strain sensitivity spectrum. Let the astrophysical model-builder integrate their favorite signal over that curve.
 - iii. In the context of some commonly-accepted model with absolutely normalized amplitudes, such as Z-M supernova waveforms, we can plot the efficiency vs the distance to the source, for a fixed S/N threshold.

Test, evaluate and optimize performance of the full analysis pipeline.

I. Prepare frame data for testing the analysis pipeline.

a. Noise models:

- i. Gaussian white noise, mean=0, SD=1.
- ii. Gaussian, but colored by expected IFO performance. Can add lines (60Hz, etc).
- iii. Results from E2E simulations (to represent, eg, a well-performing LIGO IFO circa 2003).
- iv. Real ADC data from a LIGO engineering run.

b. Signal models:

- i. Inspiral chirp. This is only an example of a kind of "burst" waveform; we must consider it as "unmodeled"; the inspiral group is applying matched templates to these.
- ii. Ringdown. Same comment as above... unless the inspiral group is NOT applying matched templates for these waveforms – in which case, WE must do that!

iii. Zwirger-Muller supernova waveforms. AJW has the 78 waveforms in their catalog (all at 10 Mpc), already pre-sampled at 16384 Hz and ready to be applied to antenna pattern and inserted into frames.

iv. Hermite-Gaussians or sine-Gaussians.

c. Apply antenna patterns for L- (and L+?) channels, for signals with randomly chosen directions and polarizations.

d. Apply IFO transfer function, to generate ADC from $h(t)$.

II. Exercise single-IFO analysis pipeline, verify performance.

a. Read in frames; apply DataCondAPI; pass data, noise spectrum, and transfer function to WrapperAPI; apply burst filters; generate triggers; insert Segment and Trigger records into database.

b. Verify that correct Segment records are produced.

c. Crudely tune thresholds to get a sensible fake trigger rate.

d. Inject large fake signals into noisy data stream; verify that Trigger records are produced with correct information.

e. Many choices here: what data segment length and overlap for the LDASjob; which classes of filters, and with what tiling; etc.

f. LDAS supplies, or we develop, a method for continually submitting jobs so as to get a large (multi-day) dataset through the system (may only need a simple Tcl script).

g. Make timing measurements: given the resources available to us, can we keep up with the data? Is there sufficient time to apply several different classes of burst filters? Should multiple filter classes be applied in the same LDAS job, or in separate jobs? Etc.

h. Run on noise + signal events, using a variety of models, and construct single-IFO fake rates vs S/N threshold, and signal efficiencies, for all classes of signals and all classes of filters that have been considered.

i. For strong signals, verify correct parameter estimation (h_{\max} or h_{rms} within filter bandwidth).

j. For all classes of signals and all classes of filters that have been considered, tune all tunable parameters to get the best possible performance (signal efficiency for fixed, tolerable, single-IFO fake rate).

k. Construct coincidence fake rates (vs. some overall S/N threshold), and signal efficiencies (for fixed S/N threshold, vs., e.g., distance), for all classes of signals and all classes of filters that have been considered.

l. Prepare report with summaries of the results, maybe even with recommendations for which burst filters are most efficient and/or estimate parameters most reliably, for which source models.

III. Combine data streams from multiple interferometers. Exercise coincidence logic. Examine choice of length of coincidence window. Evaluate false-alarm rates after coincidence.

Run full analysis pipeline on E6 data, establish upper limit on coincident event rates (or establish detection of events.)

- Ideally, this should be a fully-automated, virtually blind procedure.
- However, as pointed out by Daniel, this is unrealistic: the data stream will have quirks, non-Gaussian and non-stationary noise, burst-like events that won't be understood (at first), etc.

We must understand the calibration (IFO transfer function) thoroughly, and apply the calibration procedure sufficiently often to have it be reliable to a given level of accuracy (probably not needed to better than 10%).

- We must thus first go through the data with the goal of understanding and cleaning it up as well as possible, but without looking at the trigger rate or the triggers themselves. We should make every effort to avoid bias, either for detection or for veto.
- If signals are injected into the datastream via GDS, we can check whether they triggered our filters, and tune any appropriate cuts or procedures.
- Finally, I advocate that we do strive for a fully automated, blind analysis procedure, and live with (publish) what we've got.

Present results in an astrophysically significant way in a publication.

Summary of steps from Monte Carlo to astrophysical interpretation:

- a) Pick filter method to test.
- b) Pick a trial waveform.
- c) Pick a strength and stick to it for many trials and then adjust to map out efficiency. Or, do we want to do the astrophysical thing and pick a luminosity, then pick distances in next step, d)?
- d) Pick a random time and a random (?) place on the sky. N.B.: Special places like GC or Virgo have special dec, but random hour angle.
- e) Calculate projection of waveform onto each antenna. Straightforward with LIGO, almost as straightforward for GEO, we'll need help from ALLEGRO to make sure we do bar right. (N.B.: Need separate ALLEGRO track (from Siong?) of e) through j).)
- f) Superpose with noise (calculated or pre-recorded.) Noise time series should have built-in lock losses, and other non-stationarities that we want to model. If we want to model gain variations, need to do that in step d).
- g) Do c) through e) many times to prepare 3 time series with many signals to find.
- h) Pass time series through DMT monitors that determine lock state, as well as any others that will be queried by LDAS pipeline.
- i) Pass time series through LDAS pipeline (datacondAPI, filters.) This generates filter output versus time for each filter.
- j) Create list of threshold crossings.
How will we have set thresholds? In Monte Carlo, can look at histogram of noise without (added) signals, and know what thresholds give what probabilities (except perhaps at the 1/10 per year level, but fortunately our singles rate can be much higher and still yield coincidences at 1/10 per year.) Even with pre-recorded data, we can use coincidence information to get some confidence that no signals are present.
- k) Exercise "event logic" to establish time/filter best representing event in each time series.
- l) Check for coincidence among detectors.

How do we set threshold for coincidences? Sam/Stefano/others had suggestion on joint probability. This also feeds back into how sharply we try to do event logic, that is to say we want to check whether we falsely dismiss a coincidence if we put it in the wrong time or filter bin in one detector.

We should construct histograms of coincident event size (however we end up defining it) to investigate probability at low thresholds. We guarantee no real coincidences by using non-physical delays to generate histogram that illustrates probability distribution of false coincidences.

- m) Construct graphs of false dismissal versus event strength for interesting thresholds (in other words, efficiency.)
- n) Construct graphs of false detection (fake rate) versus threshold.
- o) Using time series without added signals and/or with time shifts, construct trial graph of rate vs. strength.

See Amaldi et al. *Astron. Astrophys.* **216**, 325 (1989). We probably want to cast the problem a bit differently than they did. Firstly, they pick a single threshold that shows some accidental coincidences, and ask what are the odds that this is drawn from the population of accidentals (as probed by the time-delay histogram.) I would think you'd want to pick a threshold with less than 1/10 per year probability of an accidental, then ask if you see a coincident signal.

Secondly, the model they test is a set of pulses all of a single strength. Why not a more astrophysically-motivated model, where events of a certain intrinsic strength are distributed homogeneously (or tracking galaxies) throughout the universe, giving a range of pulse strengths?

- p) Repeat for other filters. Comparison based on which criteria? Efficiency of detection at fixed false alarm rate, but for which signal(s)?

Comment: Translating our results to limits on any specific waveform

An arbitrary waveform ought to be characterizable by its inner product with the various filters. Best filter is the one with the largest inner product (suitably normalized as largest SNR.) For any particular waveform suggested either prior to the experiment or after the fact, we can find the best one of our filters and the ratio of that filter's performance to that of a matched filter. I think that separate upper limits for each of a large bank of filters (e.g. various center frequencies and various durations), along with this prescription of comparing an arbitrary waveform to the best-matched of our filters, is the most general thing we can say about arbitrary waveforms.

We may want to run through a large catalog of possible waveforms to explore how far we are from performance equivalent to "matched". But since the space of possible waveforms has very high dimension, it isn't possible to span the space. It isn't even clear what it would mean to sample it well.