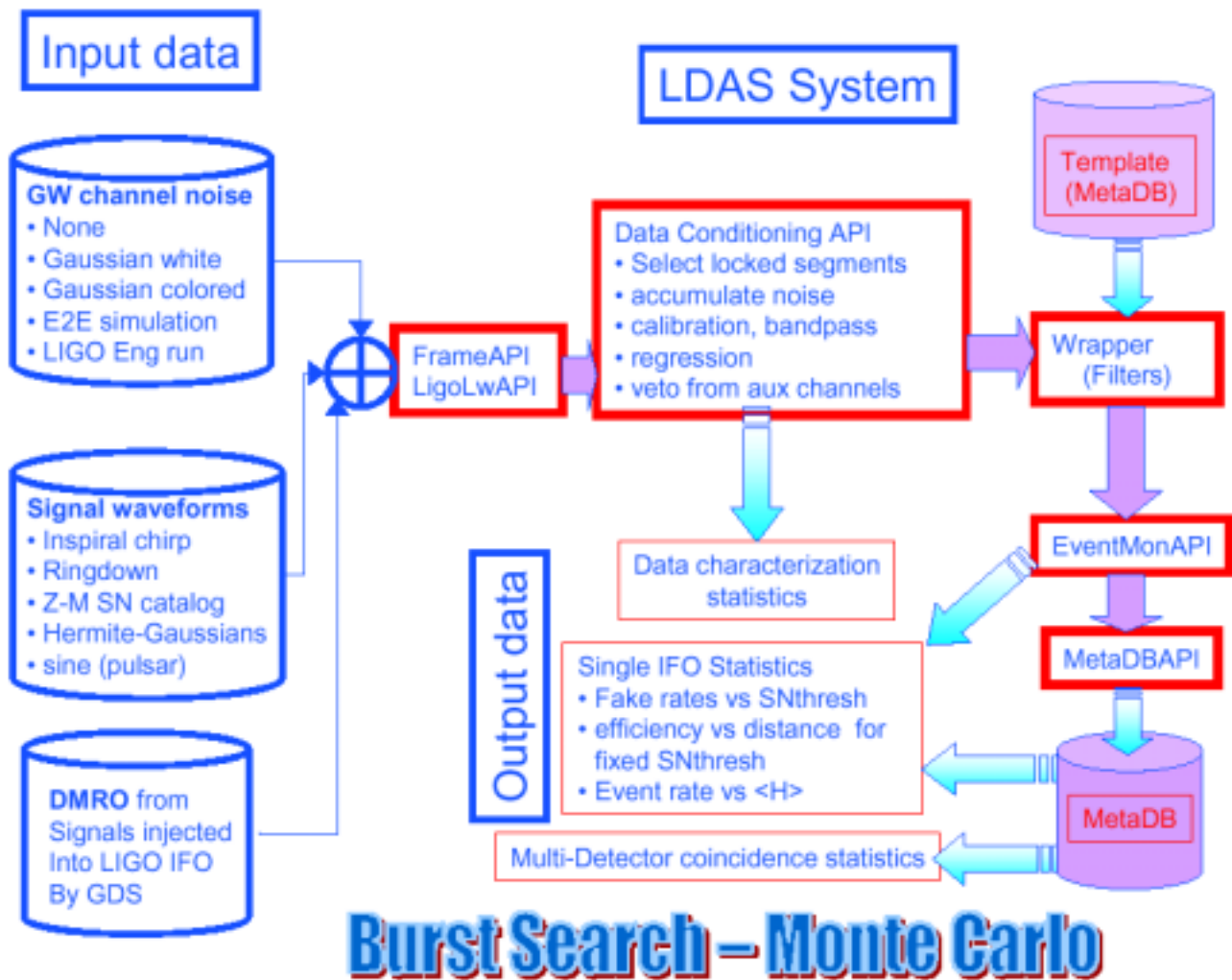


Progress on Burst Upper Limits simulation development.

AJW, 4/3/01

Here's my overview of the task at hand:



Milestones for the simulation subgroup:

Exercise LDAS user application code / pipeline

- Become familiar with the use of the ldasJob facilities
- Jan 2001 LDAS MPI Mock Data Challenge (MDC) exercised most of what we need
- Participate actively in May LDAS MDC

Dependencies:

Deliverables: results of test jobs

Schedule: 3/1 - 6/1

FTE: 0.5 * 3 mo

Preparation of simulated waveforms

burst waveforms:

- Supernova waveform menagerie from Zwerger, Muller et al
- inspiral forms such as chirps, ring-downs
- ad-hoc forms such as derivatives of gaussians
- MonteCarlo source direction/polarization/distance
- MonteCarlo datastream output h_i @ 16384Hz
- Write to data file (for e2e), frames, ilwd

Dependencies:

Deliverables: code to generate simulated signals; data files

Schedule: 3/1 - 6/1

FTE: 0.7 * 3 mo

Preparation of noise data

Noise sources for testing filters:

- no noise
- white gaussian
- colored gaussian
- E2E simulated "ideal" IFO data (limited duration).

Real data sources:

- E2 - E6 engineering data
- 40m / TAMA coincidence data
- LIGO data with GW excitation (HW -> SW test)

Dependencies:

Deliverables: code and scripts to read in data

Schedule: 4/1 - 6/1

FTE: 0.3 * 3 mo

Merging simulated signals into data

merge into e2e:

- Box to inject waveform into e2e (Han2K ETMs)
- Box to filter and write h_i @ 16384Hz
- study fidelity of output digitized stream (h_i -> DM-ADC_i)
- study effects of noise: nonlinearity, couplings with other channels

merge into data:

- Stand-alone c code to write waveforms or e2e output h_i (16384Hz) to frames, ilwd, etc
- SignalMerge: WrapperAPI Code to merge signal with data (upstream of burst filter, or even of DataConditionAPI)

Dependencies:

Deliverables: E2E read and write boxes; merge code

Schedule: 4/1 - 7/1

FTE: 0.5 * 4 mo

Assembly and preparation of code for ldas

- data conditioningAPI
- SignalMergeAPI to merge signal with data
- WrapperAPI burst filter code
 - Compile list of filter parameters, tune parameters for MC study
- MetaDataAPI code to insert event triggers into MetaDB

Dependencies: all the above code

Deliverables: ldasJobs that perform all tasks correctly with no crashes

Schedule: 4/1 - 6/1

FTE: 0.5 * 3 mo

Preparation of Coincidence code

- Coincidence code supplied by Sigg
- Stand-alone code? LDAS?
- inserts coincident event triggers into MetaDB
- generate summary plots and statistics

Dependencies: MetaDB table schemas

Deliverables: Code that reads MetaDB, generates coincident triggers, inserts into MetaDB, produces plots and statistics

Schedule: 4/1 - 6/1

FTE: 0.5 * 3 mo

Preparation of data analysis code

- single IFO fake rate vs SNR threshold for each filter
- Efficiency for source model, vs distance for fixed SNR threshold
- accuracy of parameter estimation
- event rate upper limit analysis
- many other plots and statistics

Dependencies: MetaDB table schemas

Deliverables: Code that reads MetaDatabase and produces plots and statistics

Schedule: 4/1 - 9/1

FTE: 0.5 * 5 mo (one summer student)

Exercising analysis chain on simulated data

- May MDC
- Evolving functionality
- Run on any combination of signal, noise, data source;
vary burst filters and their parameters and banks

Dependencies: everything above

Deliverables: Plots and statistics

Schedule: 4/1 - 9/1

FTE: 0.3 * 5 mo

Exploring burst filter parameter space

- optimize SNR thresholds, tune parameters
- compare filters for efficiency, parameter estimation, etc.

Dependencies: everything above

Deliverables: Plots, statistics, tables, benchmarks

Schedule: 6/1 - 9/1

FTE: 2.0 * 3 mo (two summer students)

Running analysis chain on real data

Dependencies: everything above

Deliverables: PRL

Schedule: 9/1 -

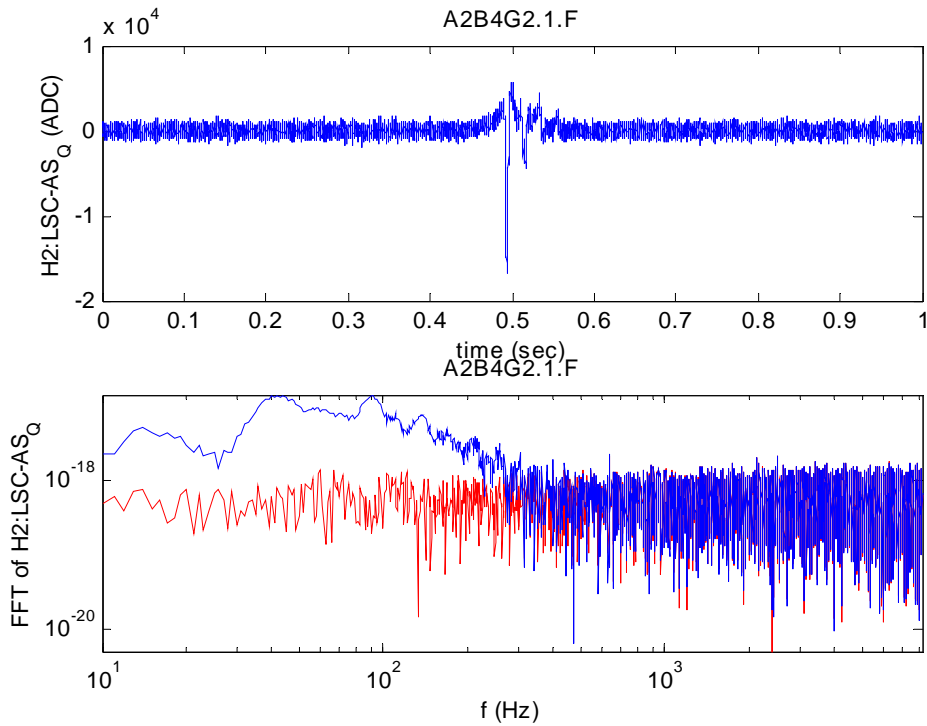
FTE: many

Current work:

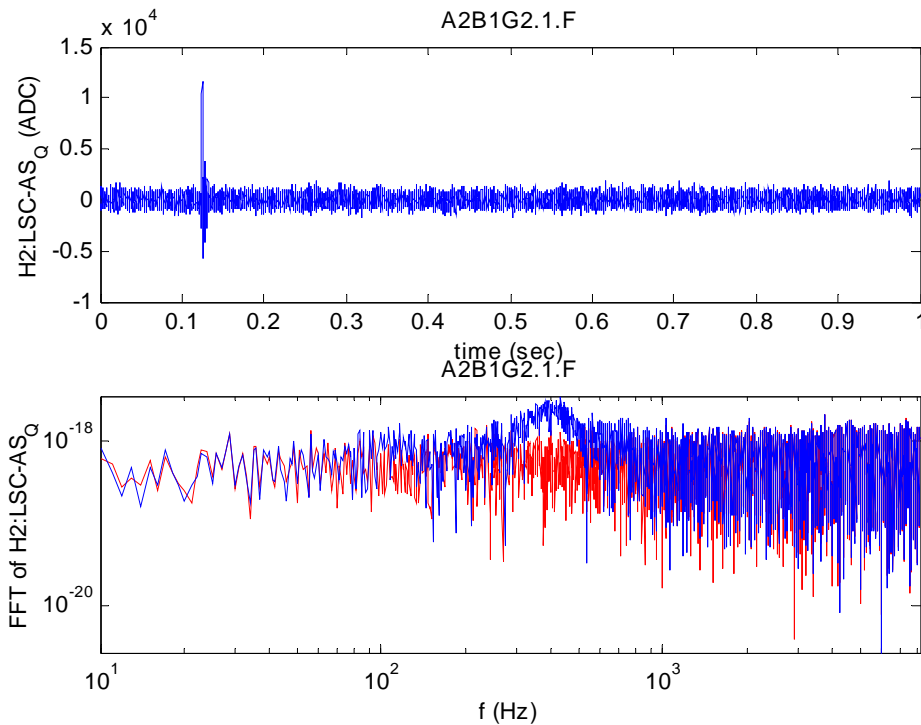
- Exercise LDAS user application code / pipeline: I can run the MPI MDC jobs in LDAS, observe their progress and their output in ilwd format. I have much to learn about ldas!
 - how to read frames with ldas (doesn't seem to work)
 - how to get ldas to get data from http (should work but doesn't)
 - how to get/use calibration data
 - how to get/use baseline noise
 - how to read/write ilwd or ldas_lw with matlab or c or root
 - how to use root to analyze data
- Preparation of simulated waveforms:
 - I have matlab code that generates ZM supernova waveforms as a function of distance; chirps as a function of mchirp and distance or h_{\max} ; ringdowns as a function of h_{\max} , f , Q ; and ad-hoc Hermite-Gaussians as a function of h_{\max} , duration, and order.
 - I can modulate by the antenna pattern, throwing random sky locations and polarization.
 - I have matlab code to resample and embed into 1-second stretches.
 - I use mkframe to write 1-second frames.
 - I do not yet know how to realistically turn $h(t)$ into ADC counts; need calibration (transfer function $h(t) \rightarrow \text{ADC}$).
 - There are lots of ways to package the data: one signal with random delay in each of many 1-second frames; many signals in many-second frames; etc. Package calibration info with the frames? Use LIGO_LW? ...
- Preparation of noise data and Merging simulated signals into data:
 - Within Matlab, I can generate random gaussian noise or read in noise from frames; and merge with the data.
 - Can use getFrames to get archived data from E2, E3, etc; or GUILD to get recent data from LHO or LLO.
 - Ususally use the 'H2:LSC-AS_Q' channel, but any specified channel(s) are possible.
 - Working on using E2E to predict noise or noise+signal.
 - Might want to do this in a c program, and/or run within LDAS?
- Assembly and preparation of code for ldas: other people's responsibility. I need something to test with, so I hope I can use the power filters that already exist in LAL, and plan to work on that soon.
- Preparation of Coincidence / data analysis code: I like the idea of using root, and hope to learn how to use it, and work with Daniel Sigg, in the near future.
- Exploring burst filter parameter space: once everything is set up and working, the fun begins.

See below for examples of signals buried in noise.

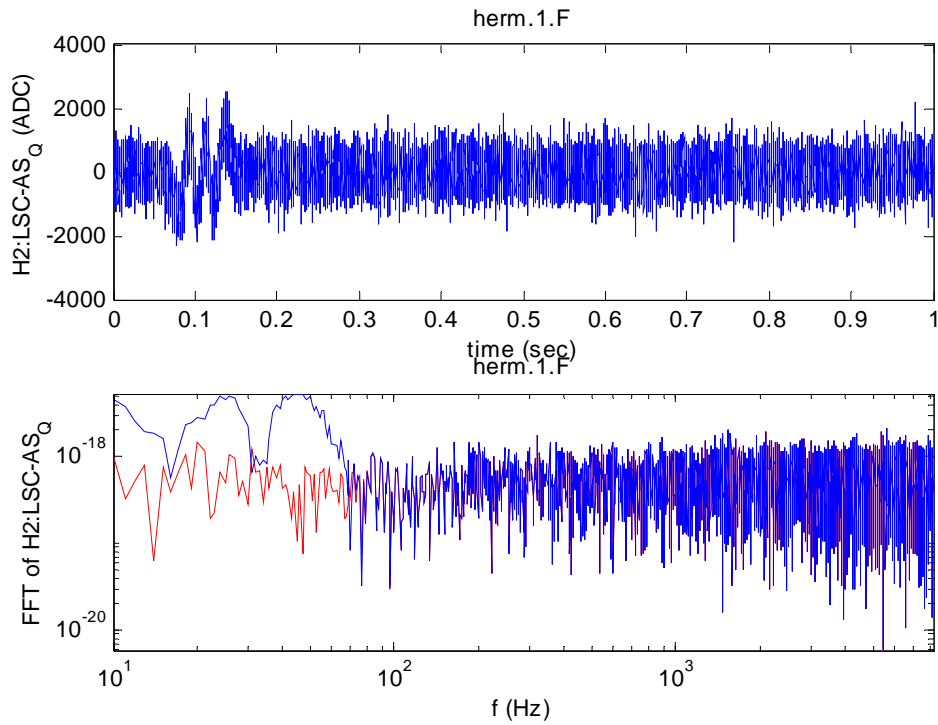
Signals buried in the noise: examples



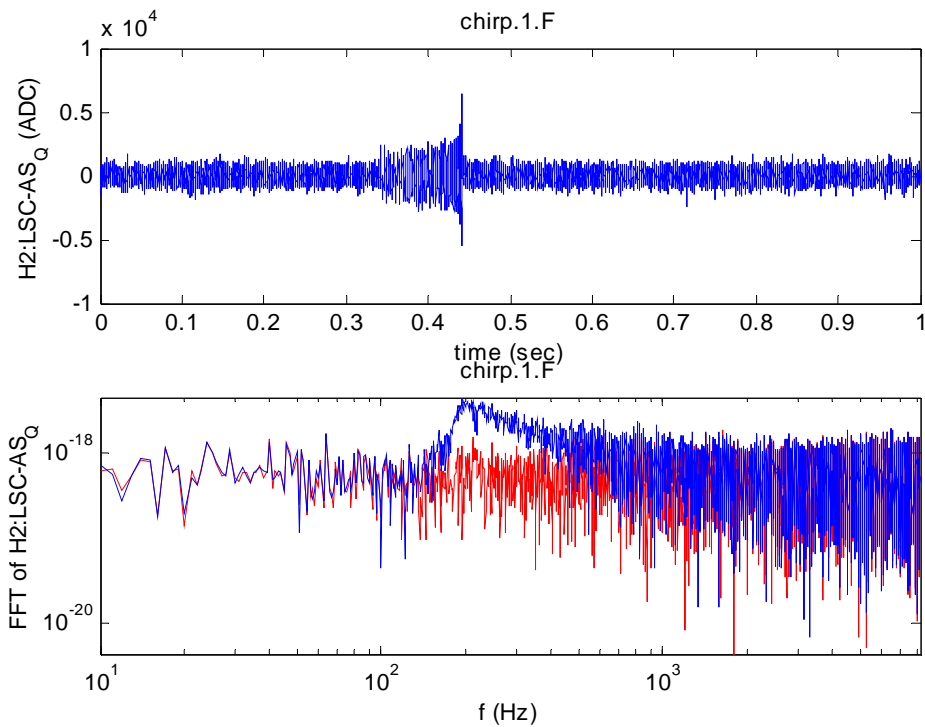
A ZM supernova waveform in white Gaussian noise. Time series, and FFT (red: noise; blue: signal+noise). The signal is introduced at a random offset from $t=0$, and is modulated by the antenna pattern in a random way. “Trivial” conversion to ADC counts.



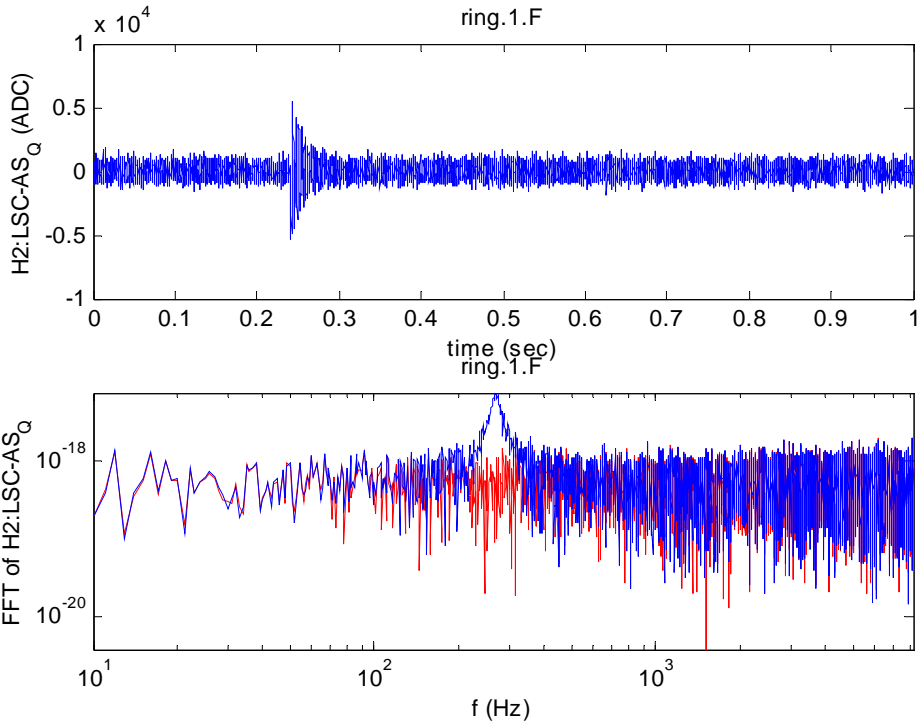
Another ZM supernova waveform in white Gaussian noise.



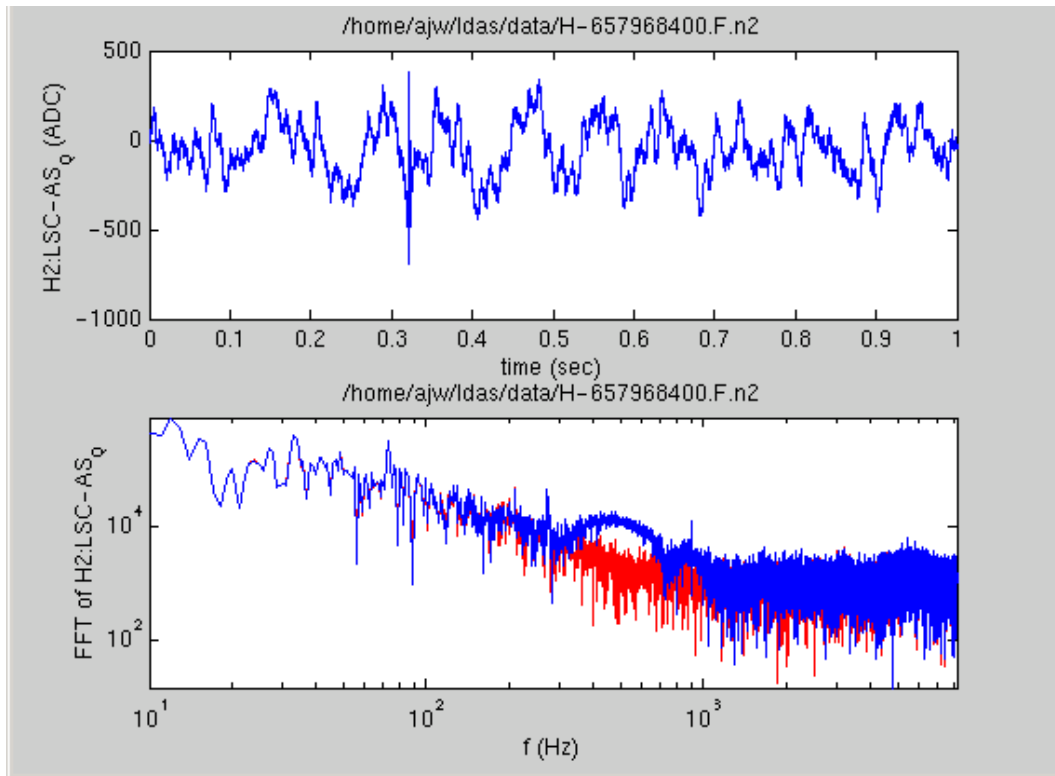
A 6^{th} -order Hermite-Gaussian on white Gaussian noise.



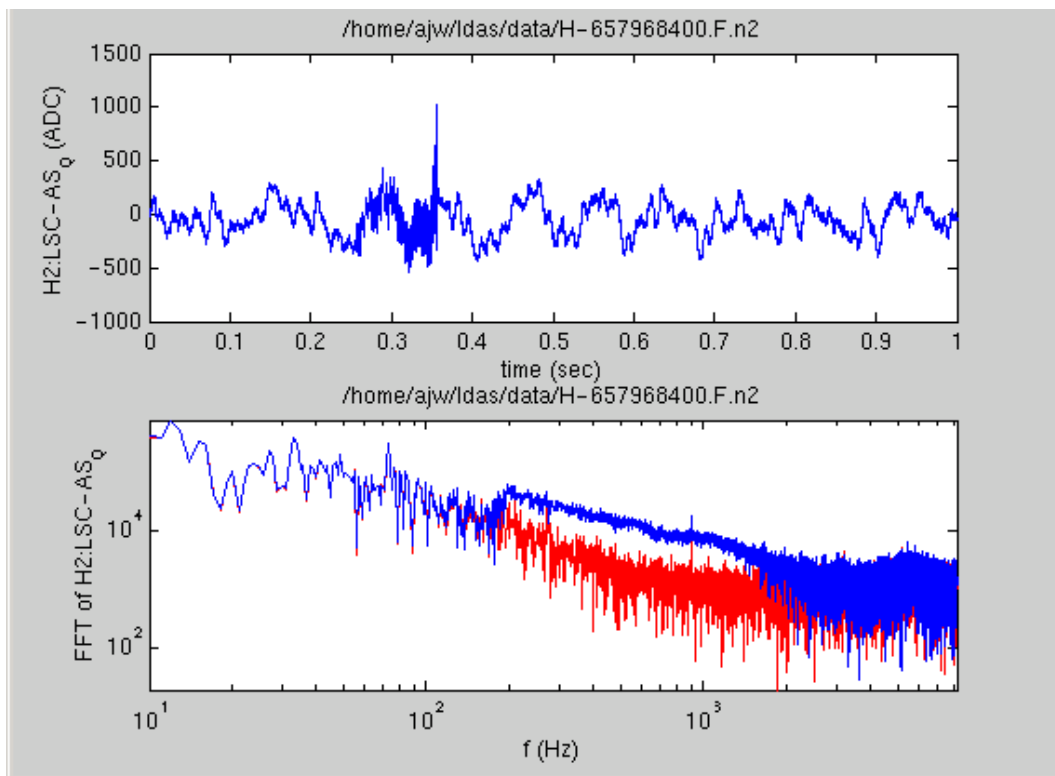
A chirp waveform on white Gaussian noise.



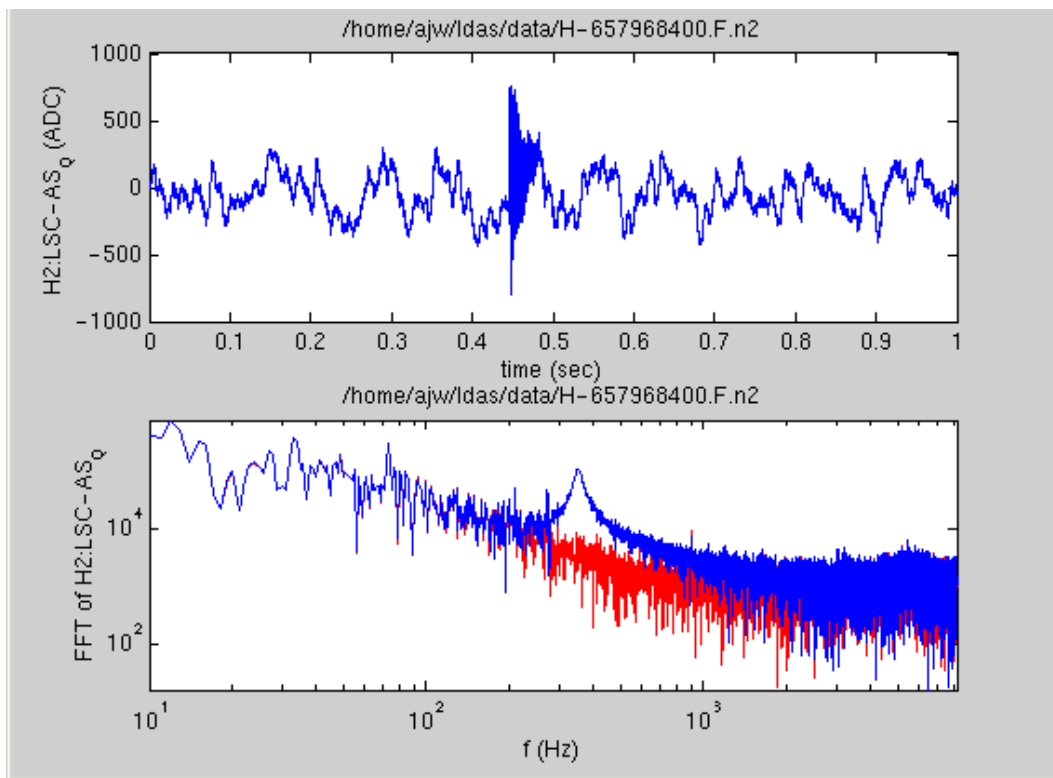
A ringdown waveform on white Gaussian noise.



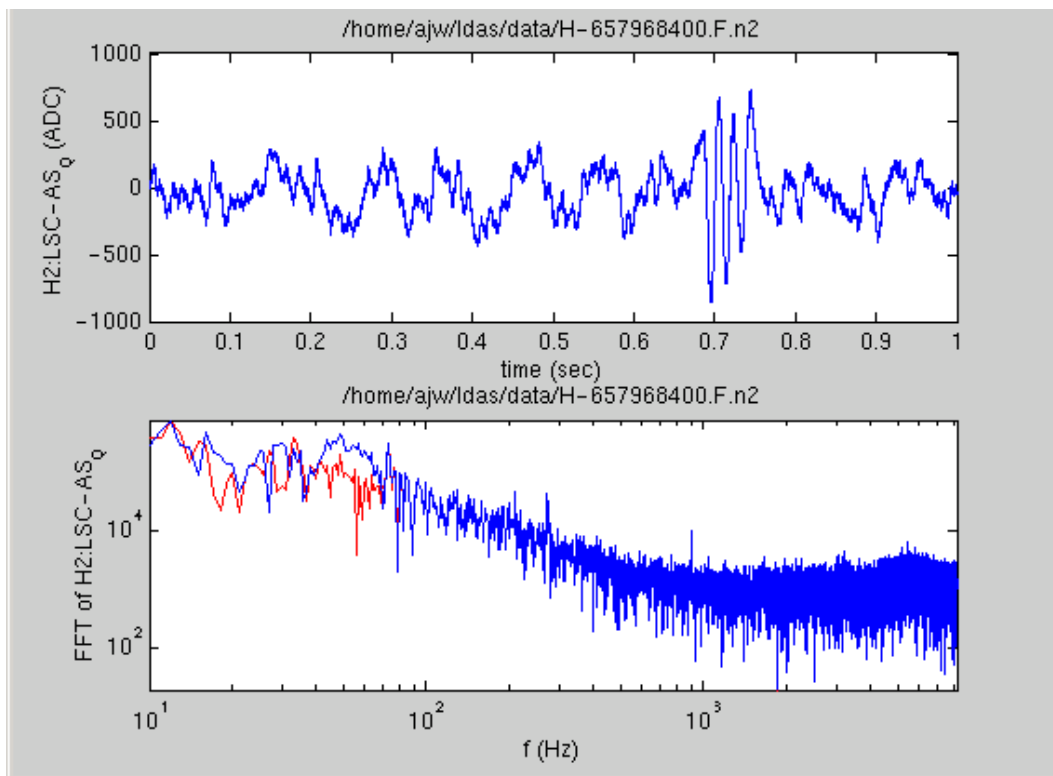
A ZM supernova waveform on some 'H2:LSC-AS_Q' data from E2 (11/11/00, 1am PST).



A chirp waveform on some 'H2:LSC-AS_Q' data from E2 (11/11/00, 1am PST).



A ringdown waveform on some 'H2:LSC-AS_Q' data from E2 (11/11/00, 1am PST).



A 6th-order H-G waveform on some 'H2:LSC-AS_Q' data from E2 (11/11/00, 1am PST).

LIGO antenna pattern

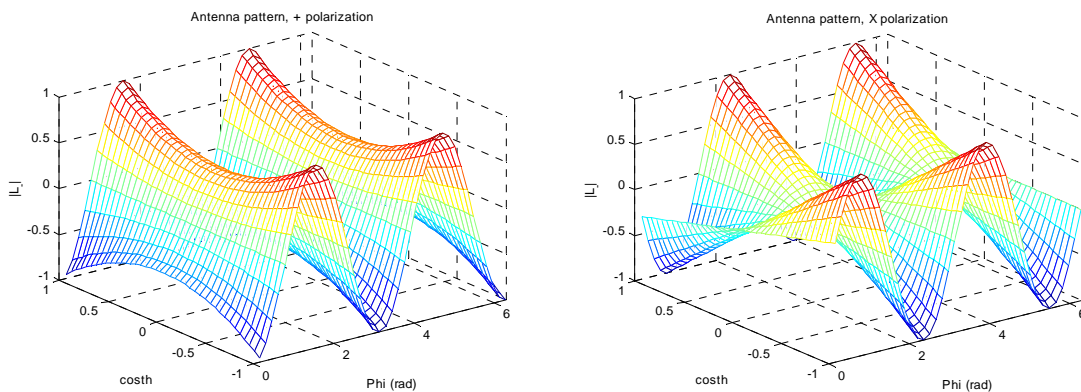
ETM response to quadrupole wave must be of the form: $dLx = x^i T^{ij} x^j$

where x^i is displacement from ITM to ETM (take to be unit x, y vectors with length L_{arm}) and T^{ij} in the “transverse traceless” gauge must be built out of the vectors characterizing the GW: the direction of motion w^i , and the direction of the GW strain perpendicular to the direction (transverse) and each other, u^i and v^i .

It must be traceless and anti-symmetric under $u \leftrightarrow v$, so it must be of the form:

$$T^{ij} = u^i u^j - v^i v^j + a (u^i v^j - v^i u^j)$$

where I don't know what a is, but it doesn't matter, since that term is zero when $x^i T^{ij} x^j$ is formed. This gives, for $x^i T^{ij} x^j / L^2$, in the $(Lx-Ly)/2$ combination:



Generated with the following code:

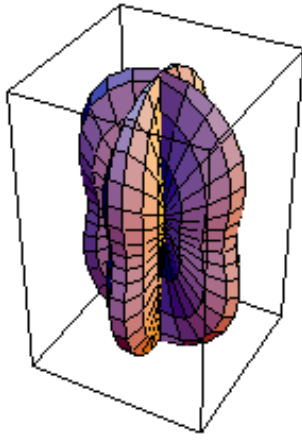
```
% construct GW direction and x/y polarization vectors
hz = [-sinh*cos(phi) -sinh*sin(phi) -costh];
wx = [-hz(2) hz(1) 0]./sqrt(hz(1)^2+hz(2)^2);
wy = cross(hz,wx);
hx = cos(psi)*wx+sin(psi)*wy;
hy = cross(hz,hx);

% convert to Lx and Ly with the antenna pattern
Lx = (hx(1)^2-hy(1)^2);
Ly = (hx(2)^2-hy(2)^2);
Lm = 0.5.*(Lx-Ly);
Lp = 0.5.*(Lx+Ly);
cLc(I,J) = Lm;
```

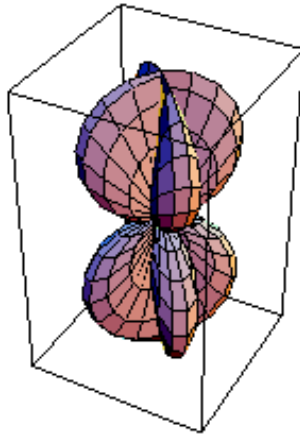
where the plot on the left corresponds to $\psi=0$, and on the right, to $\psi = \pi/4$. These patterns agree perfectly with the usual formulae for the antenna pattern:

```
% analytic form of antenna pattern
FLm = 0.5*(1+costh^2)*cos(2*phi)*cos(2*psi)
      -costh*sin(2*phi)*sin(2*psi);
FLp = sinh^2*cos(2*psi);
```

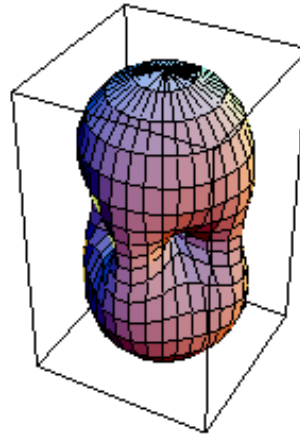
The analytic form, in Mathematica, generates the following familiar patterns (as far as I know, Matlab can't make these kind of plots):



+ polarization



× polarization



unpolarized