Discussion of Statistical Methods, Tools, and Simulations

- Review of tools and methods
- (I am only familiar with) tools and methods in use in the burst working group
  » What are the special needs of other groups?
  » What tools and methods used by other groups would be of general use?
- the simulation effort exercises and requires the entire analysis chain
Data analysis model (burst group)

- Does the model used by burst WG fit the other WGs?
  - generate fake signals, scaled to absolute $h_{\text{strain}}$
  - filter fake signals through IFO response functions
  - add to real data, or to white or colored gaussian noise
  - run search DSOs on the resulting frames
  - perform statistical analysis on results of DSO runs
  - present results in a form suitable for publication(s).
Generating simulated signals

- Generating simulated signals in units of strain is the easy and fun part!
- I think that all WGs have mechanisms for doing this.
- For burst WG:
  - Chirps
  - Ringdowns
  - Hermite-Gaussians and sine-Gaussians
  - Zwerger-Muller supernova GW simulations
Waveforms buried in E2 noise, including calibration/TF

- ZM supernova
- Chirp
- Ringdown
- Hermite-gaussian
Adding simulations to the data

- **How are fake signals added to the data, minimizing uncertainty associated with special handling of simulated data relative to real data?**
  - Make fake frame files
    - c code, matlab, ROOT tools
    - can add to data or simulated noise before making frame,
    - or can add to noise in LDAS job, "on the fly"
  - add signals to real data or noise frames in datacondAPI or frameAPI (burst group)
    - need more sequence manipulation tools in datacondAPI, so that one could add a signal from a file, at a run-time-determined time, amplitude, duration
    - could actually generate the signal in datacondAPI
    - maybe even convert from strain -> volts there as well
    - This would facilitate the accumulation of high statistics simulation runs, without having to manage huge frame files padded with zeros.
  - add signals in wrapperAPI / LAL code (inspiral group)

- **How to ensure reliable IFO response functions?**
  - Shall we standardize the filtering algorithm / mechanism?
  - I use matlab *zpk* to represent response function and *lsim* to filter the fake signal from strain -> volts. Better would be to use datacondAPI or LALWrapper.
Example DatacondAPI algorithm

-framequery { { R $site {} $times Adc($channel) }  
   { F $site /ldas_outgoing/jobs/ldasmdc_data/burst-stochastic/burstscan_e7h2.F {} Adc(0) } }  
-aliases { x = _ch0; s = _ch1; }  
-algorithms { zx = slice(x,0,5914624,1);  
   zy = slice(s,0,5914624,1);  
   zm = mul(zy,$mulsim);  
   zs = add(zx,zm);  
   zz = tseries(zs, 16384.0, $stime, 0);  
   stat = all(zz);  
   intermediate(,stat.ilwd,stat,psd of ch0);  
   pz = psd(zz,16384);  
   intermediate(,pzs.ilwd,pz,psd of ch0);  
   z = resample(zz,1,8);  
   m = mean(z);  
   y = sub(z,m);  
   q = linfilt(b,y);  
   r = slice(q,2047,737280,1); }  

Grab slice of data and simulated signal

Multiply signal by a run-time-determined multiplier,
Add to real data, make a time series

Accumulate statistics and power spectrum

Subtract mean; additional whitening
statistical analysis

For burst and inspiral:
- output of DSOs are event triggers, deposited into database
- further analysis is done at the event trigger level, although in the future, will want to look back at raw data
  » manipulate database entries
  » merge multiple entries
  » apply vetos from triggers on auxiliary channel (DMT?)
  » find coincidences
  » evaluate fake rates vs efficiencies at the same time
- Here, the Event Tool is being used
  » Sigg and Ito. OO classes in the ROOT environment for manipulating database "event" entries.
- Can also use matlab tools, including LIGOtools (Shawhan) for ingesting xml files obtained from DB queries.
- Can also use the power of DB queries.

How about the stochastic and pulsar analyses?
Other sources of fake signals

- Use of *simdata* matlab package (Finn) to generate colored noise
  - steep seismic noise spectrum makes it difficult but not impossible to generate desired spectrum

- Use of signals injected into interferometer using GDS system:
  - bursts and inspiral signals (chirps, wiggles, blips, sombrero; Gonzalez, Marka, Shawhan, during E7)
  - stochastic and line signals (Shawhan, Bose, Brown, during E7)

- How best to handle "model dependence"?
  - The burst group must struggle with model dependence because we have no good models
  - The other groups DO have good models, but will have errors and variations in those models).
LDAS and database

- What demands will be made on which LDAS installations?
  - Shall we use the site installations? Or only LSC installations devoted to this task?
  - How shall we coordinate these resources? (The LIGO Software Users Group has been discussing this)
  - How/where should data streams from LHO and LLO be combined, for analyses that require cross-correlations, like stochastic?
  - Standardized job control/bookkeeping scripts? (LSUG is developing job bookkeeping and monitoring tools, based on Shawhan’s LDASJob.)
  - How to facilitate multiple passes through the data? (Large RDS’s at LDAS installations).

- What demands will be made on which LDAS databases?
  - How best to avoid trashing the databases,
  - and how/whether to get entries from one installation to another?
  - (cf Peter Shawhan's proposals).
<table>
<thead>
<tr>
<th>Run date</th>
<th>filter</th>
<th>channel</th>
<th>times</th>
<th>mul ldas job</th>
<th>Tall</th>
<th>Tdatac</th>
<th>Tmpi</th>
<th>Ntrig</th>
<th>hrms</th>
<th>h(200-300)</th>
<th>h(300-400)</th>
<th>h(200-1000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20020314145958</td>
<td>power</td>
<td>H2:LSC-AS_Q</td>
<td>20020314145958</td>
<td>693913420-693913679</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL691</td>
<td>122.66</td>
<td>58.10</td>
<td>20.40</td>
<td>510</td>
<td>3.73e+04</td>
</tr>
<tr>
<td>20020314150231</td>
<td>power</td>
<td>H2:LSC-AS_Q</td>
<td>20020314150231</td>
<td>693913679-693913938</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL693</td>
<td>124.84</td>
<td>57.86</td>
<td>20.43</td>
<td>509</td>
<td>3.38e+04</td>
</tr>
<tr>
<td>20020314150508</td>
<td>power</td>
<td>H2:LSC-AS_Q</td>
<td>20020314150508</td>
<td>693913938-693914197</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL695</td>
<td>125.50</td>
<td>57.28</td>
<td>20.44</td>
<td>510</td>
<td>3.64e+04</td>
</tr>
<tr>
<td>20020314150739</td>
<td>power</td>
<td>H2:LSC-AS_Q</td>
<td>20020314150739</td>
<td>693914197-693914456</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL697</td>
<td>120.11</td>
<td>57.78</td>
<td>19.43</td>
<td>510</td>
<td>4.09e+04</td>
</tr>
<tr>
<td>20020314151010</td>
<td>power</td>
<td>H2:LSC-AS_Q</td>
<td>20020314151010</td>
<td>693914456-693914715</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL699</td>
<td>121.36</td>
<td>59.06</td>
<td>19.33</td>
<td>514</td>
<td>3.50e+04</td>
</tr>
<tr>
<td>200203150516</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203150516</td>
<td>693960121-693960481</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL121</td>
<td>172.62</td>
<td>84.36</td>
<td>33.27</td>
<td>104</td>
<td>1.20e+07</td>
</tr>
<tr>
<td>200203150817</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203150817</td>
<td>693960481-693960841</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL125</td>
<td>113.53</td>
<td>34.20</td>
<td>33.22</td>
<td>524</td>
<td>4.83e+03</td>
</tr>
<tr>
<td>200203150934</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203150934</td>
<td>693960841-693961201</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL127</td>
<td>110.86</td>
<td>32.41</td>
<td>33.35</td>
<td>391</td>
<td>7.66e+03</td>
</tr>
<tr>
<td>200203151034</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151034</td>
<td>693961201-693961561</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL129</td>
<td>169.09</td>
<td>97.74</td>
<td>31.53</td>
<td>158</td>
<td>8.01e+03</td>
</tr>
<tr>
<td>200203151138</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151138</td>
<td>693961561-693961921</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL131</td>
<td>202.59</td>
<td>88.10</td>
<td>33.25</td>
<td>12</td>
<td>6.47e+03</td>
</tr>
<tr>
<td>200203151206</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151206</td>
<td>693961921-693962281</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL134</td>
<td>120.87</td>
<td>35.83</td>
<td>31.21</td>
<td>514</td>
<td>9.45e+04</td>
</tr>
<tr>
<td>200203151321</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151321</td>
<td>693962281-693962641</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL136</td>
<td>109.86</td>
<td>35.02</td>
<td>32.26</td>
<td>505</td>
<td>6.26e+03</td>
</tr>
<tr>
<td>200203151437</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151437</td>
<td>693962641-693963001</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL138</td>
<td>107.89</td>
<td>33.26</td>
<td>32.31</td>
<td>452</td>
<td>4.74e+03</td>
</tr>
<tr>
<td>200203151534</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151534</td>
<td>693963001-693963461</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL140</td>
<td>110.00</td>
<td>33.42</td>
<td>33.28</td>
<td>477</td>
<td>4.52e+03</td>
</tr>
<tr>
<td>200203151644</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151644</td>
<td>693963461-693963821</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL142</td>
<td>112.97</td>
<td>35.30</td>
<td>32.34</td>
<td>575</td>
<td>4.07e+03</td>
</tr>
<tr>
<td>200203151754</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151754</td>
<td>693963821-693964081</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL144</td>
<td>111.41</td>
<td>34.16</td>
<td>33.27</td>
<td>465</td>
<td>3.55e+03</td>
</tr>
<tr>
<td>200203151854</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151854</td>
<td>693964081-693964441</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL146</td>
<td>171.81</td>
<td>101.1</td>
<td>31.29</td>
<td>181</td>
<td>1.23e+07</td>
</tr>
<tr>
<td>200203151954</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203151954</td>
<td>693964441-693964801</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL148</td>
<td>174.27</td>
<td>99.61</td>
<td>33.24</td>
<td>500</td>
<td>8.76e+07</td>
</tr>
<tr>
<td>200203152032</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203152032</td>
<td>693964801-693965161</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL150</td>
<td>110.73</td>
<td>35.90</td>
<td>32.56</td>
<td>605</td>
<td>4.18e+03</td>
</tr>
<tr>
<td>200203152135</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203152135</td>
<td>693965161-693965521</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL152</td>
<td>100.62</td>
<td>30.72</td>
<td>32.25</td>
<td>2</td>
<td>7.82e+05</td>
</tr>
<tr>
<td>200203152235</td>
<td>tfcluster</td>
<td>L1:LSC-AS_Q</td>
<td>200203152235</td>
<td>693965521-693965881</td>
<td>0e0</td>
<td>mit</td>
<td>NORMAL154</td>
<td>154.24</td>
<td>84.77</td>
<td>32.27</td>
<td>154</td>
<td>1.21e+07</td>
</tr>
</tbody>
</table>

LDAS job bookkeeping
Modest proposal

- Collect useful tools and pubs in a web page
- Establish mailing list for people interested in developing or using these tools
- Coordinate with LSUG (and LDAS, datacond, etc)
Progress report on Burst Simulation

Alan Weinstein, Caltech, 3/20/02

- Burst waveforms
- Calibration
- E7 data
- LDAS jobs
- Results from TFClusters
- More work to be done
Burst waveforms

- Start with simple, easy to interpret waveforms: damped sinusoids have well-defined central frequency and bandwidth:
  \[ h(t) = h_{\text{peak}} \exp\left(-\frac{t}{\tau}\right) \sin(2\pi f_{\text{cent}} t), \quad \text{BW} = 1/\tau \]
- Choose narrow bandwidth for now, \( \tau = 0.1 \text{ sec}, \text{BW} = 10 \text{ Hz} \)
- Scan over range of \( f_{\text{cent}}, h_{\text{peak}} \)
- Consider other bandwidths, other waveforms, later.
- Since we’re analyzing lots of data (~512 secs) per job, inject multiple waveforms in one job, so that we don’t have to run so many jobs…
- BUT, if these waveforms are BIG, and if the DSO calculates average power using the data itself, many injected waveforms could throw it off…
- For now, this is just a convenience…
32 waveforms, each 2 sec long, Scanning from 50 to 1600 Hz in 50 Hz steps.

The first 2 waveforms, with $f_{\text{cent}} = 50$ and 100 Hz; $\tau = 0.1$ sec

ASD of this 64-sec stretch of simulated data.

Spectrogram to illustrate the frequency scan.
E7 data

- Want to run at MIT
- GUILD reports that at MIT, we have
  - 693960000 693967184 H R gwf /export/E7/LHO/frames
  - 693960000 693967184 L R gwf /export/E7/LLO/frames
- These are 2 hrs of data from 1/1/02, when all 3 IFOs are in lock.
- This is not playground data. We need playground data at MIT.
- In the meantime, I choose a 361-sec stretch, since TFCLUSTERS apparently likes to run on that much data (I need to learn how to change that, if possible): 693961586-693961946, H2:LSC-AS_Q. (This stretch has lots of noise bursts).
Injecting bursts

- The burst signals are absolutely normalized by $h_{\text{peak}}$. Need to put it into same units as H2:LSC-AS_Q (volts) by using response function, obtained from calibration.
- The burst signals are passed through a linear filter implementing the E7 H2 calibration transfer function, then saved to a frame file and ftp’ed to
  > http://www-ldas.mit.edu/ldas_outgoing/jobs/ldasmdc_data/burst-stochastic/burstscan_e7h2.F
- Add signals to the data in LDAS DatacondAPI; can scale magnitude of signals as desired, at run-time.

```plaintext
-framequery { { R H {} $times Adc($channel) } }  
{ F H /ldas_outgoing/jobs/ldasmdc_data/burst-stochastic/burstscan_e7h2.F {} Adc(0) } } 
-aliases { x = _ch0; s = _ch1; }  
-algorithms { 
  zx = slice(x,0,5914624,1);  
  zy = slice(s,0,5914624,1);  
  zm = mul(zy,1,e0);  
  zs = add(zx,zm);  
  zz = tseries(zs, 16384,0, $stime, 0);  
  pz = psd(zz,16384);  
  intermediate(pzs.ilwd,pz,psd of ch0);  
  z = resample(zz,1,8);  
  m = mean(z);  
  y = sub(z,m);  
  q = linfilt(b,y);  
  r = slice(q,2047,737280,1); }
```
E7 calibration

http://blue.ligo-wa.caltech.edu/engrun/Calib_Home/

<table>
<thead>
<tr>
<th>Run</th>
<th>Gain</th>
<th>Poles (Hz)</th>
<th>Zeros</th>
<th>Channel to use</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2 (LHO 2k)</td>
<td>6.8e-8</td>
<td>0.74, 0.74, 0, 0</td>
<td>50.35, 50.35, 12.1, 12.1, 186</td>
<td>E2 LSC-AS_0 Q</td>
</tr>
<tr>
<td>E3 (LLO one arm)</td>
<td>4.9e-6</td>
<td>0.74, 0.74, 0.1, 0.1, 1000</td>
<td>7.38, 7.38, 7.38, 313, 313</td>
<td>L1 LSC-AS_1</td>
</tr>
<tr>
<td>E4 (LLO two arms)</td>
<td>1.69e-9</td>
<td>0.74, 0.74, 0, 0</td>
<td>16.82, 0.391, 0.391, 232.6, 232.6</td>
<td>L1 LSC-AS_0 Q</td>
</tr>
<tr>
<td>E7 LHO 2k fully recycled</td>
<td>1.89e-8</td>
<td>0.74, 0.74, 0, 0</td>
<td>119.5, 33.04, 33.04, 33.04, 33.04</td>
<td>E2 LSC-AS_0 Q</td>
</tr>
<tr>
<td>E7 LHO 4k recombined</td>
<td>6.00e-8</td>
<td>0.74, 0.74, 0, 0</td>
<td>20.28, 20.28, 20.28, 20.28, 20.28</td>
<td>E1 LSC-AS_0 Q</td>
</tr>
</tbody>
</table>

Except for the double pole at 0.74 Hz (mirror pendulum) the poles and zero are the result of a transfer function fit.

No calibration info from LLO has been posted here yet.
Add bursts to data

Time series

Calibrated strain noise spectrum

Noise spectrum

Ratio of noise spectra, With/without injected signal

AJW, Caltech, LSC Meeting, 3/20/02
**BIG Bursts added to E7 data**

(as a check)

Noise ASD for 361 secs of H2:LSC-AS_Q (red), And with BIG bursts added during secs 201-264 (blue).
Bursts are added, and PSDs obtained, using LDAS/DataCond (thanks to Philip Charlton for his help).

Strain sensitivity from 361 secs of H2:LSC-AS_Q (red), And with BIG bursts added (blue).
Note that red curve is in good qualitative agreement with spectrum in Calib page, and bursts scan frequencies from 50-1600 Hz in 50 Hz steps, bandwidth = 10 Hz, and all with same peak strain.

**AJW, Caltech, LSC Meeting, 3/20/02**
DSO search

- Run with TFCLUSTERS, 361 seconds at a time.
- Run on 361 sec data segment from H2, no injected signals:
  - 357 triggers into mit_test sngl_bursts table.
- Inject 32 bursts with $h_{\text{peak}} = 1 \times 10^{-16}$, scanning $f_{\text{cent}}$ from 50 to 1600 Hz in 50 Hz steps, signals spaced 2 secs apart, starting at sec 200.
  - 471 triggers into mit_test sngl_bursts table.

Most big SNR triggers are unchanged after injection of simulated bursts. Many of the first 16 bursts stand out over the fakes.
Efficiencies: Presenting the results

Find loudest trigger within 1 sec of injected burst. Plot SNR vs frequency of injected burst.

Note that accidental coincidence of injected burst with noise burst obscures injected bursts at, eg, 350, 500, 550, 850, 1000 Hz.

Compare SNR of triggers coincident with injected bursts, with measured noise spectrum. Arbitrary relative scale, for now – needs work!

Anyway, it looks like with the burst amplitudes that were injected, we run out of efficiency above ~1100 Hz.
Power DSO

- Running on 260 sec stretches of playground E7 data
- With no signals injected, get 510 triggers (hard limit??)
- If I run with large signals, baseline power (calculated from the same data stretch that we are searching in!) gets trashed;
  - ALL snrs go down for ALL triggers.
  - Even in the windows where signals are injected.
- With smallish signals injected, still get 510 triggers, but they do seem to show up a bit.
- Still, with these huge numbers of large SNR bursts, how can we hope to see signals that should be seen given the mean power levels?
SLOPE DSO

- Ran on 260 secs of E7 data from 1/1/02
- With no signals injected, get 5938 triggers
- With signals injected, get same (?) 5938 triggers
- At the moment, can’t seem to run slope DSO anymore at MIT; get wrapperAPI errors that data are unavailable…
First look at L1

- 361 secs of L1:LSC-AS_Q (693961586-693961946) around 1/1/02.
- Hmm. Doesn’t look a lot like expected…
More work

- Get the right playground data at MIT.
- Run on colored gaussian noise with same PSD as data.
- Get absolute scale right.
- Consider all three IFOs (currently, LLO calibration not available).
- Learn how to tune/optimize DSOs.
- Consider other bandwidths, waveforms.
- Learn how to use Event class in ROOT. (Currently use MATLAB).
- Enhance DatacondAPI capabilities to more easily modify the injected bursts on the fly.
- Automate LDAS submissions, Trigger processing (*rundso* script).
- Decide on best way to summarize results.