LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	**LIGO-T040020-01-Z**	2004/05/19
GravEn Simulation Engine Primer		
Amber L. Stuver		

Distribution of this draft:

Circulation open to all LSC members

DOCUMENTATION FOR VERSION 1.0

California Institute of Technology
LIGO Project - MS 51-33
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: http://www.ligo.caltech.edu/

Processed with LaTeX on 2004/05/19

Contents

List of Figures

1 GravEn Overview

GravEn (Gravitational-wave Engine) simulates gravitational wave signals that can be added into the data stream of a gravitational wave detector. Every aspect of the signal can be specified, such as maximum strain, location on the sky, when the signal starts (including inter-sample start time), etc.

The general operation of GravEn is summarized in section 3.2. **All of the needed calculations dealing specifically with the detector are in the coordinate system that is centered on the origin of the WGS-84 geodetic model of the Earth.** This gives a common coordinate system to multiple detectors so that coincident simulations between multiple IFO's can be produced. The end result is a timeseries in units of AS_Q counts that can be directly added to existing data from a detector.

This primer is specifically for the MATLAB version of this simulation engine. While the compiled version will work in the same way, methods of variable input will be slightly different than those described here. GravEn does not require any MATLAB toolboxes to function properly.

Please read the `README.txt` files that accompany the source code for GravEn and its test suite.

GravEn has undergone PSURG code review[1]. For further information and to submit bug reports, contact Amber Stuver at stuver@gravity.psu.edu.

2 Variable Names

Throughout this document, function names and computer input and output are represented in `fixed width font`. Variable names, as described in the following, are *italicized*:

- *ampl* - maximum strain of simulations; can be scalar, 1×2 or 1×3 vector (q.v. section 4.2)
- *detId* - string identifying detector for which simulation is being made
- *External* - structure[2] containing source's location on sky and polarization angle
  - *.x* - cosine of the source's co-declination sky location as projected onto Earth's fixed coordinates
  - *.phi* - source's sky location longitude as projected onto Earth's fixed coordinates (in rad)
  - *.psi* - source's polarization angle (in rad), q.v. section 7.3 for complete definition
- *gps* - GPS start time (in whole seconds)
- *inputFileName* - (string) name of file containing information for generating simulations (can be used as a cell for pre-generated waveforms, q.v. section 3.2.2)
- *Internal* - structure[2] containing description of source's orientation to line-of-sight

[1] See LIGO-T040035-00-Z for the description of PSURG MATLAB coding standard.

[2] Internal and External can be input as different classes (i.e. structure, double or string) however, GravEn will convert to and use them as a structure (q.v. section 4.2).

  - *.x* - cosine of the angle the line-of-sight makes with the source's axis of angular momentum
  - *.phi* - angle between the source's X-axis and the projection of the line-of-sight on the rotation plane, measured counterclockwise (in rad)

- *logFileName* - (string) name of log file
- *nSim* - number of simulations to generate
- *sampFreq* - sampling frequency of detector (in Hz)
- *seed* - state for random number generator
- *simId* - string containing information about type of simulation to be generated; **Very format dependent (q.v. section 5.2)**
- *startSamp* - range of allowed start times (or determined start time after the list function) in samples with respect to *gps*

3 Driver Function - `graven`

3.1 Inputs/Modes of Operation

There are several modes of operation for GravEn. The mode is determined by the number of variables input into the driver function.

For simulations that generate original waveforms, modes of input are:

- 6 inputs - the simulation information is read line-by-line from a file

  ```
  graven(logFileName,gps,detId,inputFileName,sampFreq,seed)
  ```

- 7 inputs - the simulation information is read by making *nSim* random draws on the lines of a file

  ```
  graven(nSim,logFileName,gps,detId,inputFileName,sampFreq,seed)
  ```

- 11 inputs - the driver takes the simulation information directly from the inputs and does not need a file to read

  ```
  graven(nSim,logFileName,gps,detId,simId,ampl, ...
                          startSamp,Internal,External,sampFreq,seed)
  ```

The syntax to push a pre-generated waveform through GravEn is:

- 9 inputs

  ```
  graven(nSim,logFileName,gps,detId,inputFileName, ...
                                  startSamp,external,sampFreq,seed)
  ```

3.2 Structure of the Driver

3.2.1 Generate Original Waveforms

First, the driver decides what mode to operate in based on the number of input variables and then assigns the variables accordingly. The appropriate variables are then used as input into the list function for the generation of the master simulations list (each line contains the information for one simulation).

With the list made, the driver initiates a loop to make simulations by going line-by-line through the master simulations list. For each simulation, the information from one line of the simulations list is read and variables assigned.

Knowing the source location on the sky and which detector this simulation uses, the driver calls a function that calculates the appropriate number of samples to add to the start time to account for the difference in arrival time of a gravitational wave between the center of the Earth and the detector.

The driver then calls a function to create the desired metric perturbation and then feeds this perturbation into the TT gauge projector. With the metric perturbation in the TT gauge, the + and $\times$ polarizations are returned and then fed into the detector projector. This projects the gravitational wave on the antenna pattern of the detector at the center of the Earth.

The detector projector returns the timeseries in units of strain. This must then be converted into units of AS_Q counts so that the simulation may be added to the data stream. This is done using the inverse calibrator function (it is inverse in that the original calibration is determined to convert AS_Q counts into strain, and we are interested in converting strain into AS_Q counts). The final timeseries in units of AS_Q counts is then saved in a structure along with the start time of the simulation at the detector. Information for the simulation is saved to a log file such that the log file can be used as the input file for the list function to recreate the waveform. The loop then increments and goes through the next line of the master simulations list.

3.2.2 Push Pre-generated Waveforms

The structure of the driver is exactly the same for pre-generated waveforms except that, once the pre-generated waveform is read from file, the metric perturbation creation function and the TT gauge projector are bypassed along with rounding the start time of the simulation at the detector to the nearest sample, since inter-sample start times for pre-generated waveforms are not yet supported.

Currently, pre-generated waveforms can be read from ilwd-files and text files. *inputFileName* has a special mode of operation when used for pre-generated waveforms. The file name can be input as a string or a cell array. If *inputFileName* is a 1×1 cell array, it is treated as a string; if it is a 1×2 cell array, the first element is treated as the string input file name and the second element is the field path where the time series can be found (appropriate for ilwd-files). An example 1×2 *inputFileName* cell for a mat-file is:

```
{'inputFileName', 'fieldPath'}
```

If a 1×1 cell array is input, and the input file is a ilwd-file, the field path for the time series is

assumed to be `'m1.real_8'`[3].

Several assumptions are made about the pre-generated waveform:

- The waveform is in the TT gauge
- The waveform contains + polarization only
- The waveform is evenly sampled at the same rate as *sampFreq*[4]

It is important to make sure that the *sampFreq* input into GravEn matches that of the pre-generated waveform and the *detId* for accurate results.

3.3 Outputs

The output of `graven` is a vector of structures:

- *Sim* - vector of structures containing the simulations
  - *.name* - the *simId* string
  - *.nstr* - the start time of the simulation in whole samples at the detector
  - *.sig* - the simulation time series in units of AS_Q counts
  - *.gps0* - the start time of the simulation in seconds and nanoseconds at the detector
  - *.pkgps* - the time of maximum strain in seconds and nanoseconds at the detector

4 List Function - `makelist`

4.1 Modes of Operation

The list function's modes of operation are directly associated with those of the driver function:

- 1 input - generate the simulations list from file line-by-line

  `makelist(inputFileName,detId` *`(optional)`*`)`
- 2 inputs - generate simulations list from *nSim* random draws on the lines of a file

  `makelist(nSim,inputFileName,detId` *`(optional)`*`)`
- 6 input - generate simulations list from inputs directly

  `makelist(nSim,simId,ampl,startSamp,Internal, ...`

  `External,detId` *`(optional)`*`)`

[3]The `'m1.real_8'` field path is the path used within the pre-generated waveform ilwd-files found on the LIGO Burst Group Simulations page at http://www.ligo.caltech.edu/∼ajw/bursts/burstsim.html.

[4]e.g. The original Ott, et al (astro-ph/0307472) waveforms are not evenly sampled, but waveforms that have been filtered to be evenly sampled at 16384 samples per second can be found on the LIGO Burst Group Simulations page[3].

4.2 Modes of Input

Several variables have their own mode of operation that this function handles:

- *ampl*

  The amplitude in units of strain can be input in one of three ways when *ampl* is directly input into the list function (no file reading):

  - scalar - this is a fixed amplitude
  - 1×2 vector - the amplitude is randomly chosen from a logarithmic distribution in the range *ampl*(1):*ampl*(2)
  - 1×3 vector - the amplitude is randomly chosen from the even linear distribution *ampl*(1):*ampl*(2):*ampl*(3)

- *Internal* and *External*

  Internal and *External* contain the angles needed to describe the source, as far as rotations are concerned. *Internal* has two angles and *External* has three, as described in section 2.

  When reading from a file, any or all of these angles can, instead of having a numerical value, be the string `'RANDOM'` or `'OPTIMAL'`[5]. In the case that an angle is `'RANDOM'`, the needed angle is chosen from the appropriate range; if an angle is `'OPTIMAL'`, the angle is chosen as described in the following paragraph (external angles can only be `'OPTIMAL'` if the optional *detId* is input into `makelist`).

  When *Internal* and *External* are passed directly into the list function and no files are needed, *Internal* can be a structure or 1×2 vector containing the numerical values for the internal angles, *External* can be a structure or 1×3 vector containing the numerical values for the external angles or either *Internal* or *External* can be the string `'RANDOM'` or `'OPTIMAL'`. In the case that *Internal* and/or *External* is the `'RANDOM'` string, all of the internal angles and/or all of the external angles must be randomly chosen from within allowed ranges. In the case that *Internal* is the `'OPTIMAL'` string, it will be chosen so that the source's axis of angular momentum is coincident with the line-of-sight (i.e. no rotations are needed). In the case that `'OPTIMAL'` is chosen for *External*, the source's sky location is chosen to be zenith to the detector and the polarization angle is 22.5^o (for equal weighting of + and $\times$ polarizations). Thus, *detId* needs to be input into the list function only when *External* is chosen to be `'OPTIMAL'`. If *detId* is not input and `'OPTIMAL'` is input for *External*, the list function will return an error and stop the driver function.

4.3 File Formats

When the list function needs to read from a file, the file must be a text file and have the following format:

```
simId ampl startSamp1 startSamp2 int1 int2 ext1 ext2 ext3
```

[5] `'RANDOM'` and `'OPTIMAL'` may be in any form of mixed case.

where the range of allowed start times (in samples) is given by `startSamp1` to `startSamp2` (lower to upper), internal angles *Internal.x* and *Internal.phi* are `int1`, and `int2`, and the external angles *External.x*, *External.phi* and *External.psi* are `ext1`, `ext2`, and `ext3`. The file may be space and/or tab delimited and any information in the tenth column or higher is ignored (such as when using a log file from previous simulations as *inputFileName*). N.B. An input file cannot contain a combination of simulations whose waveforms are originally generated in GravEn and pre-generated waveforms.

Sample file for original waveforms:

```
SG~512~0.01~0.25 1e-21 10 20 -0.707 +0.7941 +0.426 +0.349 -1.6755
CG_525_0.015_0.2 1e-21 100 200 RANDOM Random random rAnDoM RaNdOm
G_000~0.02_0.4 1e-23 1024 16384 Random +0.6590 random -1.7453 RANDOM
BH~530_0.01~0.2 1.25e-20 1024 4096 0.001 -2.1817 +0.5001 1.0472 1.1345
```

Sample file for pre-generated waveforms:

```
./preGen/ZM_A1B2G1.ilwd,m1.real_8 -1 1 163840 1 0 0.743343 1.491 4.766
./preGen/ZM_A1B1G2.ilwd -1 100 200 RANDOM Random random rAnDoM RaNdOm
./preGen/ZM_A3B3G5.ilwd -1 1024 16384 Random +0.6590 random -1.7453 RANDOM
./preGen/ZM_A3B2G1.txt -1 1024 4096 0.001 -2.1817 +0.5001 1.0472 1.1345
```

Note that the amplitude for pre-generated waveforms is always -1. This is a flag for GravEn to call the appropriate listing function (q.v. section 10, `pregenlist`) and change the structure of the driver (q.v. section 3.2.2). Since GravEn knows nothing about the source, it assumes that the internal angles are at optimal orientation. If internal angles are specified through the input file (as they are above), they will be ignored.

The *simId* value can contain both components of the cell used to specify the file name and field path of pre-generated waveforms as seen in the *simId* element of the first row. The values are separated by a comma **without any spaces**. If a space is included with the comma, `makelist` will produce an error since it will read the second value separate from the first (i.e. the space will be read as a delimiter).

Population simulations are intended to be manifest in the input file. GravEn does not generate populations other than homogeneous, isotropic populations by specifying angles to be `'RANDOM'`.

4.4 Outputs

The output is a cell array containing the master simulations list. Each row is a simulation:

```
simId ampl startSamp int1 int2 ext1 ext2 ext3
```

where `startSamp` is now a scalar containing the determined start time in samples.

5 Metric Perturbation Function - `makeh`

5.1 Inputs

- *simId*
- The fixed signal amplitude from `makelist`
- The start time in samples (only interested in the decimal part for the inter-sample start time)
- *sampFreq*

5.2 *simId* Format

The *simId* string is very format dependent. This string contains the identifier for the type of simulation (e.g. 'CG' for cosine-Gaussian, 'G' for Gaussian), the frequency of the waveform, the standard deviation of the involved Gaussian (τ) and the duration of the simulation in seconds (Gaussians are centered on the length of the timeseries). The format for this is:

`'type_frequency_`τ`_duration'` (this string may also be $\sim$ delimited)

e.g. `'SG_530_0.01_0.25'`[6] will yield a sine-Gaussian with a frequency of 530 Hz, a τ of 0.01 and a timeseries length equal to one quarter of a second[7].

Another function is called to parse this string and return the separate values in a form usable by the code (string for the type and double for the frequency, τ and duration); q.v. section 10, `simparse`.

5.3 Currently Supported Waveforms

Currently supported waveforms are Gaussian (`'G'`), sine-Gaussian (`'SG'`), cosine-Gaussian (`'CG'`), sinc (`'SINC'`) and black hole ringdown (`'BH'`):

$$G(t,\tau) = ampl * e^{-(\frac{t}{\tau})^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{1a}$$

$$SG(f,t,\tau) = ampl * \sin(2\pi f t) e^{-(\frac{t}{\tau})^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{1b}$$

$$CG(f,t,\tau) = ampl * \cos(2\pi f t) e^{-(\frac{t}{\tau})^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{1c}$$

[6] *simId* my be in any form of mixed case.

[7] N.B. Since frequency is not defined for Gaussians, it does not matter what frequency is entered in the *simId* string since the frequency will not be used in the metric perturbation calculation; for waveforms that do not involve Gaussians (e.g. `'SIN'`, `'SINC'`), the entered τ does not matter.

$$SINC(f,t) = ampl * \frac{\sin(\pi f t)}{\pi f t} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (1d)$$

$$BH(f,t,\tau) = ampl * e^{-\frac{t}{\tau}} \begin{pmatrix} \sin(2\pi f t) & \cos(2\pi f t) & 0 \\ \cos(2\pi f t) & \sin(2\pi f t) & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (1e)$$

In addition to these waveforms, sine waves (`'SIN'`), cosine waves (`'COS'`), linear chirp[8] (`'CHIRP'`), square waves (`'SQUARE'`), Gaussian modulated square waves (`'SQUAREG'`), sawtooth waves (`'SAWTOOTH'`) and Gaussian modulated sawtooth waves (`'SAWTOOTHG'`) can be generated.

Please see the current help file for an up-to-date list of supported types of simulations.

5.4 Outputs

- The 3×3×time array containing the time evolution of the metric perturbation specified in the *simId* string

6 TT Gauge Projector - `makett`

6.1 Inputs

- The metric perturbation in its full 3×3×time array format
- *Internal*

6.2 Calculation

The source is defined to be in a right-handed Cartesian coordinate system with the rotation of the source in the XY plane and the angular momentum in the Z direction ($\hat{z} = \hat{x} \times \hat{y}$). *Internal.x* describes the projection of $\hat{n}$ along +$\hat{z}$ and $cos(Internal.phi)$ describes the projection of $\hat{n}$ along +$\hat{x}$ ($\hat{n}$ points in the direction of the line-of-sight, from the source to the Earth), q.v. figure 1.

Rotate the source's coordinate system so that $\hat{z}$ coincides with $\hat{n}$ by rotating about $\hat{z}$ an angle φ counterclockwise and then rotating about the new $\hat{y}$ counterclockwise an angle θ:

$$\begin{pmatrix} \hat{e}_x \\ \hat{e}_y \\ \hat{n} \end{pmatrix} = \begin{pmatrix} \cos(\varphi)\cos(\theta) & \sin(\varphi)\cos(\theta) & -\sin(\theta) \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ \cos(\varphi)\sin(\theta) & \sin(\varphi)\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} \quad (2)$$

where φ is *Internal.phi* and $\theta = cos^{-1}(Internal.x)$.

[8] The linear chirp begins at 0 Hz and sweeps up to the frequency specified through *simId*.

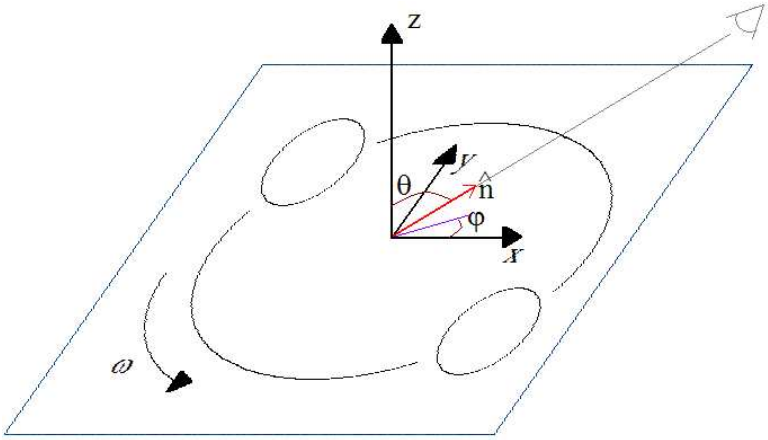

Figure 1: Illustration of source internal angles

The + and × bases are constructed by the following:

$$\hat{e}_+ = \hat{e}_x \otimes \hat{e}_x - \hat{e}_y \otimes \hat{e}_y \tag{3a}$$
$$\hat{e}_\times = \hat{e}_x \otimes \hat{e}_y + \hat{e}_y \otimes \hat{e}_x \tag{3b}$$

h_+ and $h_\times$ are then formed by contracting the metric perturbation by the appropriate basis:

$$h_+ = \frac{1}{2} h^{ij} \hat{e}_{+ij} \tag{4a}$$
$$h_\times = \frac{1}{2} h^{ij} \hat{e}_{\times ij} \tag{4b}$$

6.3 Outputs

- The 2×time vector containing the + and × polarizations:

  (1,:) - + polarization
  (2,:) - × polarization

7 Detector Projector - `detproj`

7.1 Inputs

- The 2×time vector containing the + and × polarizations of the gravitational wave
- *detId*
- *External*

7.2 Calculation

The detector is assumed to be located at the center of the Earth as defined by the WGS-84 coordinate system ($\hat{Z}$ pointing toward the North Pole, $\hat{X}$ pointing toward the intersection of the Prime Meridian and the Equator, and $\hat{Y} = \hat{Z} \times \hat{X}$) and the gravitational wave is assumed to be a plane wave incident to the North Pole.

Rotate Earth's axes so that $\hat{Z}$ points towards the source along the line-of-sight:

$$\begin{pmatrix} \hat{e}_x \\ \hat{e}_y \\ \hat{n} \end{pmatrix} = \begin{pmatrix} \cos(\varphi)\cos(\theta) & \sin(\varphi)\cos(\theta) & -\sin(\theta) \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ \cos(\varphi)\sin(\theta) & \sin(\varphi)\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{pmatrix} \tag{5}$$

where θ and φ are the source's sky location in Earth's co-declination ($\theta = cos^{-1}(\textit{External.x})$) and longitude (*External.phi*) as projected on the sky.

Since both the source and Earth Z axes now lie on the line-of-sight, we can define the + and $\times$ polarization basis in the source frame:

$$\hat{e}_+ = \hat{e}_x \otimes \hat{e}_x - \hat{e}_y \otimes \hat{e}_y \tag{6a}$$

$$\hat{e}_\times = \hat{e}_x \otimes \hat{e}_y + \hat{e}_y \otimes \hat{e}_x \tag{6b}$$

Now define the + and $\times$ polarization basis in the detector frame:

$$\hat{e}'_+ = \cos(2\psi)\hat{e}_+ + \sin(2\psi)\hat{e}_\times \tag{7a}$$

$$\hat{e}'_\times = \cos(2\psi)\hat{e}_\times - \sin(2\psi)\hat{e}_+ \tag{7b}$$

where ψ is the polarization angle *External.psi*. See section 7.3 for definition of the ψ angle.

The antenna projection matrix is formed by:

$$D = \hat{V} \otimes \hat{V} - \hat{W} \otimes \hat{W} \tag{8}$$

where $\hat{V}$ and $\hat{W}$ are the unit vectors for the X and Y arms of the detector in the WGS-84 coordinate system.

Define the beam pattern functions to be:

$$F_+ = \frac{1}{2} D^{ij} \hat{e}'_{+\,ij} \tag{9a}$$

$$F_\times = \frac{1}{2} D^{ij} \hat{e}'_{\times\,ij} \tag{9b}$$

The final detector projection is:

$$h = F_+ h_+ + F_\times h_\times \tag{10}$$

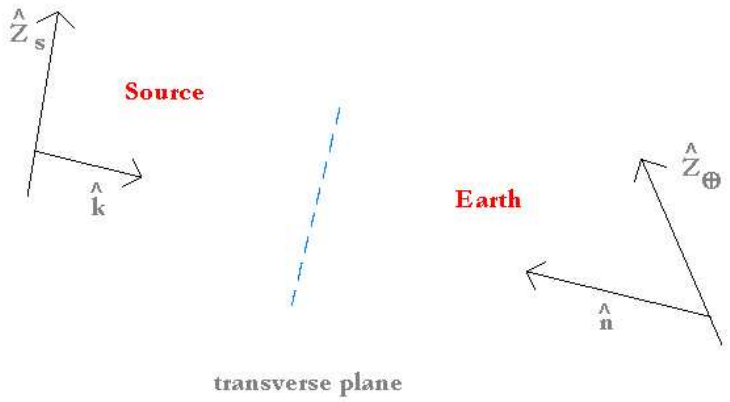

Figure 2: Definition of the transverse plane

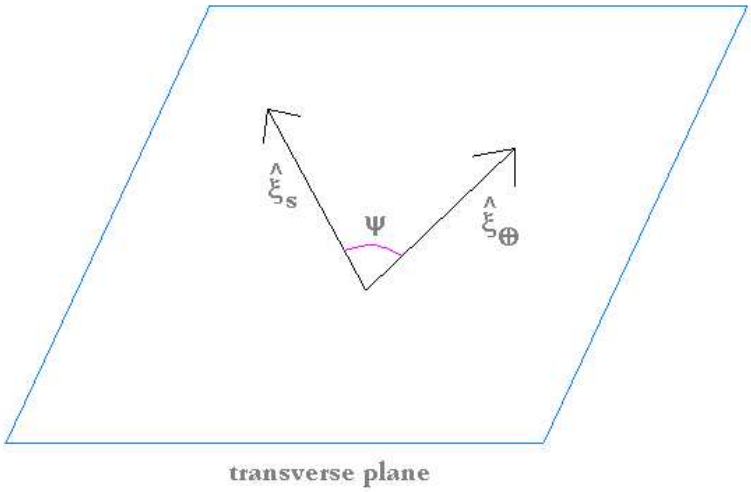

Figure 3: Definition of ψ

7.3 Definition of the ψ Angle

Define a coordinate plane between the Earth and the source that is perpendicular to the line-of-sight. This is the transverse plane of the TT gauge. $\hat{n}$ is defined to be the unit vector pointing from the center of the Earth to the source and $\hat{k}$ is defined to be the unit vector pointing from the center of mass of the source to the Earth, q.v. figure 2.

Determine the projection of the source's Z-axis and the Earth's Z-axis onto the transverse plane, $\vec{\xi}_s$ and $\vec{\xi}_\oplus$ respectively:

$$\vec{\xi}_s = \hat{z}_s - \hat{k}(\hat{k} \cdot \hat{z}_s) \tag{11a}$$

$$\vec{\xi}_\oplus = \hat{Z}_\oplus - \hat{n}(\hat{n} \cdot \hat{Z}_\oplus) \tag{11b}$$

ψ is defined to be the angle between $\vec{\xi}_s$ and $\vec{\xi}_\oplus$, q.v. figure 3:

$$\psi = \cos^{-1}\left[\frac{\hat{Z}_\oplus - \hat{n}(\hat{n}\cdot\hat{Z}_\oplus)}{|\hat{Z}_\oplus - \hat{n}(\hat{n}\cdot\hat{Z}_\oplus)|}\cdot\frac{\hat{z}_s - \hat{k}(\hat{k}\cdot\hat{z}_s)}{|\hat{z}_s - \hat{k}(\hat{k}\cdot\hat{z}_s)|}\right] \tag{12}$$

Since:

$$\hat{k}\cdot\hat{z}_s = Internal.x \tag{13a}$$
$$\hat{n}\cdot\hat{Z}_\oplus = External.x \tag{13b}$$
$$\hat{k} = -\hat{n} \tag{13c}$$

ψ is then:

$$\psi = \cos^{-1}\left[\frac{\hat{Z}_\oplus - \hat{n}(External.x)}{|\hat{Z}_\oplus - \hat{n}(External.x)|}\cdot\frac{\hat{z}_s + \hat{n}(Internal.x)}{|\hat{z}_s + \hat{n}(Internal.x)|}\right] \tag{14}$$

When the Earth's Z-axis is parallel or anti-parallel to the line-of-sight, ψ is measured as the angle between $\hat{X}_\oplus$ and $\vec{\xi}_s$:

$$\psi = \cos^{-1}\left[\hat{X}_\oplus\cdot\frac{\hat{z}_s + \hat{n}(Internal.x)}{|\hat{z}_s + \hat{n}(Internal.x)|}\right] \tag{15}$$

When the source's Z-axis is parallel or anti-parallel to the line-of-sight, ψ is measured as the angle between $\vec{\xi}_\oplus$ and $\hat{x}_s$:

$$\psi = \cos^{-1}\left[\frac{\hat{Z}_\oplus - \hat{n}(External.x)}{|\hat{Z}_\oplus - \hat{n}(External.x)|}\cdot\hat{x}_s\right] \tag{16}$$

When both the Earth's and the source's Z-axes are parallel or anti-parallel to the line-of-sight, ψ is measured as the angle between $\hat{X}_\oplus$ and $\hat{x}_s$:

$$\psi = \cos^{-1}\left[\hat{X}_\oplus\cdot\hat{x}_s\right] \tag{17}$$

7.4 Outputs

- The simulated waveform timeseries in units of strain

8 Inverse Calibrator - `calibsimfd`

8.1 Inputs

- *detId*
- *gps*

- The simulation timeseries in units of strain
- *sampFreq*
- An optional cache to hold filter coefficients

8.2 Calibrations

In LIGO, calibrations are measured to convert the AS_Q counts into strain. Here, we are interested in converting strain into AS_Q counts. In this sense we are performing an inverse calibration.

The response function essentially represents the calibration function at a given time. LIGO calculates the response function in units of strain/AS_Q counts:

$$R(f,t) = \frac{1 + \alpha(t)\beta(t)H(f)}{\alpha(t)C(f)} \tag{18}$$

where $H(f)$ is the complex open loop gain, $C(f)$ is the complex sensing function, α is the optical gain and β is the DARM gain[9].

As can be seen here, the response function is easily separated into the time dependent sum of two fixed filters:

$$\frac{1 + \alpha(t)\beta(t)H(f)}{\alpha(t)C(f)} = \frac{1}{\alpha(t)C(f)} + \frac{\beta(t)H(f)}{C(f)} \tag{19}$$

The inverse calibration cannot be separated so easily:

$$\frac{\alpha(t)C(f)}{1 + \alpha(t)\beta(t)H(f)} \neq \frac{\alpha(t)C(f)}{1} + \frac{C(f)}{\beta(t)H(f)} \tag{20}$$

This property has created difficulties in producing the inverse calibration in the time domain. However, the frequency domain calibration has proven effective with the caveat that it cannot be done 'on-the-fly.'

8.3 Frequency Domain Inverse Calibrator

Doing the inverse calibration is rather straightforward. First, the strain data is Fourier transformed using the FFT algorithm, then multiplied by the response function in units of AS_Q/strain:

$$R^{-1}(f,t) = \frac{\alpha(t)C(f)}{1 + \alpha(t)\beta(t)H(f)} \tag{21}$$

and the resultant is inverse Fourier transformed back into the time domain. The time domain series is now in units of AS_Q counts.

[9] See LIGO-T030097-00-D for a description of LIGO calibration

8.4 File Formats

The inverse calibrator reads H, C, α and β from files for the appropriate times for the simulation. These files are taken directly from the calibration data on calibrations team's web page at:

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

Format of the $\alpha\beta$ file

Each row contains the α and β data for a specific time period:

```
GPS alpha*beta alpha beta CalLine/Ref
```

Sample $\alpha\beta$ file:

```
Time alpha*beta alpha beta CalLine/Ref
729273600 0.9191546569916 0.9191546569916 1 0.9287903263751
729273660 0.9717164296205 0.9717164296205 1 0.9752651307225
729273720 0.914159856745 0.914159856745 1 0.9243393296843
729273780 0.913896049417 0.913896049417 1 0.9241040761594
729273840 0.9981778508986 0.9981778508986 1 0.9984121955427
```

The naming convention for these $\alpha\beta$ files is:

```
<detId>AlphaBeta.txt
```

An example file name is as follows:

```
H1AlphaBeta.txt
```

Format of the open loop gain ($H(f)$) and sensing function ($C(f)$) file

Since $H(f)$ and $C(f)$ are complex functions of frequency, each row in the file contains:

```
frequency magnitude phase
```

so that

$$H(f) \;\&\; C(f) = magnitude(f) * e^{i*phase(f)} \tag{22}$$

Sample open loop gain ($H(f)$) or sensing function ($C(f)$) file:

```
1.0000000e+000 1.5535765e+008 1.8015259e-001
2.0000000e+000 5.0997246e+006 3.8203867e-002
3.0000000e+000 9.7043534e+005 -1.7550441e-002
```

```
4.0000000e+000 3.1206339e+005 -9.6032949e-002
5.0000000e+000 1.2880833e+005 -1.9426868e-001
⋮
```

The file names contain the time for which the data represents. The naming convention is:

```
<detId>_olg_GPStime.txt
```

for the open loop gain and

```
<detId>_sf_GPStime.txt
```

for the sensing function. An example file name is as follows:

```
H1_olg_734234126.txt
```

8.4.1 Comment on Location of Calibration Files

`calibsimfd` needs to be told where to find the calibration files. Find the line of code in `calibsimfd` that reads:

```
DEFAULT_CALIB_DIR = '/usr/center/raid1/s2/calibration';
```

which at the time of this writing is on line 48. Replace this file location with the location appropriate for the system.

8.4.2 Comment on Using `calibsimfd` for Data Runs Other Than S2

`calibsimfd` needs to be told the names of the calibration file for the particular data run (it is currently programmed for the LIGO data run S2). Find the lines of code in `calibsimfd` that read:

```
if(IFO == 'H1')
   olgFile = 'H1_olg_734073939.txt';
   sfFile = 'H1_sf_734073939.txt';
else
   if(IFO == 'H2')
      if(gpsSec < 731849043)
      olgFile = 'H2_olg_734234126.txt';
      sfFile = 'H2_sf_734234126.txt';
   else
      olgFile = 'H2_olg_734234127.txt';
      sfFile = 'H2_sf_734234127.txt';
   end
else
```

```
   if(IFO == 'L1')
      olgFile = 'H2_olg_734234126.txt';
      sfFile = 'H2_sf_734234126.txt';
   end
   end
end
```

which at the time of this writing are on lines 110 through 128. Replace these file names with the appropriate file names for the data run.

The same $\alpha\beta$ file name can be used for future data runs as long as the new coefficients are appended to the old coefficients.

8.5 Outputs

- The simulation time series in units of AS_Q counts (*Sim.sig*)
- The cache for filter coefficients

9 Format of Log File

The log file serves to record all of the information used to produce a simulation. Therefore, each line of the log file contains the following information:

```
[simId ampl startSamp1 startSamp2 Internal.x Internal.phi ...
              External.x External.phi External.psi refGPS ...
                          gps0sec gps0ns pkampl pkgps_sec pkgps_ns detId]
```

The contents of the log file are chosen to allow the file to be recycled into the `makelist` function. In accordance with that, `startSamp1` must equal `startSamp2` and are in units of samples with respect to `refGPS` (`refGPS`=*gps*), which is the simulation start time at the center of the Earth. `gps0sec` and `gps0ns` is the start time of the simulation at the IFO (*detId*) in seconds and nanoseconds. `pkampl` is the maximum strain of the simulation and `pkgps_sec` and `pkgps_ns` is the time of the maximum strain at the IFO (*detId*) in seconds and nanoseconds. All other entries are the same as the variables of the same name used for the simulation.

There are also two header lines in the log file indicated by the first character of #. The first header line states the time the simulation was initiated, and the second labels the content of the following rows. The log file is space delimited.

Log File From Original Waveforms

```
# This simulation was run on 5/18/2004 at 10:38:19.15 EDT
# (simId) (ampl) (startSamp1) (startSamp2) (Internal.x) (Internal.phi) (External.x) (External.phi) (External.psi) (gps) (gps0 s) (gps0 ns) (pkampl) (pkgps s) (pkgps ns) (detId)
SG~512_0.01~0.2 1e-21 3676.706 3676.706 9.261102e-01 0 7.854000e-01 1.311747e-01 3.926991e-01 729273660 729273660 217755198 1.724996e-22 729273660 317810058 H1
G~000~0.015~0.3 1e-19 596.966 596.966 7.070000e-01 3.927000e-01 -6.472000e-01 2.356200e+00 6.936000e-01 729273660 729273660 49314856 -4.248406e-45 729273660 349243164 H1
BH_600~0.25_0.5 1e-23 132.7972 132.797 9.218496e-01 -1.434441e+00 9.638590e-01 1.611192e+00 2.434623e+00 729273660 729273659 996581077 5.339607e-24 729273659 997375488 H1
CG~512_0.01~0.2 1e-21 5102.784 5102.784 -2.599272e-01 0 7.854000e-01 2.289409e+00 3.926991e-01 729273660 729273660 302357912 3.156001e-22 729273660 402404785 H1
BH_600~0.25_0.5 1e-23 165.265 165.265 -7.777317e-01 -1.614092e+00 1.663284e-01 1.805366e+00 1.770585e+00 729273660 729273660 18088936 3.854974e-24 729273660 19714355 H1
g~000~0.015~0.3 1e-19 912.765 912.765 7.070000e-01 3.927000e-01 -6.472000e-01 2.356200e+00 6.936000e-01 729273660 729273660 68589687 -4.213877e-45 729273660 368530273 H1
SINC~000~0.015~0.3 1e-19 805.091 805.091 7.070000e-01 3.927000e-01 -6.472000e-01 2.356200e+00 6.936000e-01 729273660 729273660 62017798 -4.099878e-45 729273660 361999511 H1
BH_600~0.25_0.5 1e-23 160.515 160.515 6.138684e-01 1.982067e+00 9.423621e-02 -1.895397e+00 4.994582e+00 729273660 729273659 993931531 5.301686e-24 729273659 993957519 H1
G~000~0.015~0.3 1e-19 706.271 706.271 7.070000e-01 3.927000e-01 -6.472000e-01 2.356200e+00 6.936000e-01 729273660 729273660 55986285 -4.129960e-45 729273660 355957031 H1
SG~512_0.01~0.2 1e-21 14300.243 14300.243 -8.457130e-01 0 7.854000e-01 2.802236e+00 3.926991e-01 729273660 729273660 859131932 6.230189e-22 729273660 958190917 H1
```

Log File From Pre-generated Waveforms

```
# This simulation was run on 5/18/2004 at 11:24:10.32 EDT
# (simId) (ampl) (startSamp1) (startSamp2) (Internal.x) (Internal.phi) (External.x) (External.phi) (External.psi) (gps) (gps0 s) (gps0 ns) (pkampl) (pkgps s) (pkgps ns) (detId)
./preGen/ZM_A3B3G5.ilwd,default -1 15664.682 15664.682 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273660 953552246 5.257746e-24 729273660 984313964 H1
./preGen/ZM_A3B3G5.ilwd,default -1 10437.659 10437.659 1 0 5.670902e-01 -1.745300e+00 1.794641e+00 729273660 729273660 616882324 1.326986e-23 729273660 647644042 H1
./preGen/ZM_A1B2G1.ilwd,m1.real_8 -1 11414.295 11414.295 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273660 694091796 6.916681e-23 729273660 788757324 H1
./preGen/ZM_A1B2G3.ilwd,m1.real_8 -1 4709.561 4709.561 1 0 5.606721e-01 1.311747e-01 5.943829e+00 729273660 729273660 286071777 7.979139e-24 729273660 381958007 H1
./preGen/ZM_A1B1G2.ilwd,default -1 158.369 158.369 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273660 7141113 3.128788e-23 729273660 76721191 H1
./preGen/ZM_A1B2G1.ilwd,m1.real_8 -1 46797.678 46797.678 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273662 853759765 6.916681e-23 729273662 948425292 H1
./preGen/ZM_A1B1G2.ilwd,default -1 112.667 112.667 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273660 4333496 3.128788e-23 729273660 73913574 H1
./preGen/ZM_A1B2G1.ilwd,m1.real_8 -1 50474.732 50474.732 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273663 78186035 6.916681e-23 729273663 172851562 H1
./preGen/ZM_A3B3G5.ilwd,default -1 2223.676 2223.676 1 0 7.433430e-01 1.491000e+00 4.766000e+00 729273660 729273660 133178710 5.257746e-24 729273660 163940429 H1
./preGen/ZM_A2B1G2.ilwd,default -1 123.323 123.323 1 0 -6.786640e-01 1.624412e+00 4.001315e+00 729273660 729273660 26977539 4.608877e-23 729273660 96557617 H1
```

10 Other Needed Functions

- `ctextread` - reads formatted data from text file; modified from MATLAB built-in function `textread` to avoid the use of `which` (used to make GravEn compilable)
- `fileext` - utility to parse a string containing a file name into the directory path, name and file extension
- `getifo` - retrieves the IFO coordinates and arm unit vectors for the given *detId* (all LIGO information taken from LIGO-P000006-D-E; other IFO's (VIRGO, TAMA, etc.) to be added in a future version)
- `h2asq_fd` - performs the inverse calibration on the simulated strain data as described in section 8.3
- `ifodelay` - calculates the number of samples to add to the selected start time of a simulation to account for the gravitational wave's flight time, in the direction of propagation, from the center of the Earth to *detId* (the number of samples can be negative)
- `ilwdread` - load an ILWD (Internal LightWeight Data) formatted XML file
- `ndraws` - function to perform random draws on the lines of *inputFileName*
- `ndx2gps` - converts samples to time (in whole seconds and nanoseconds) after the reference *gps* time in seconds
- `pickstime` - pick the start time of the simulation from a range of samples
- `pregenlist` - alternate listing function to `makelist` for use with pre-generated waveforms
- `randomangle` - randomly choses a random angle within the appropriate range for the angle type
- `read_alphabetafile` - reads the $\alpha\beta$ file and returns α and β corresponding to the GPS time of the simulation
- `setampl` - pick the waveform's amplitude from the appropriate distribution
- `setangles` - driver function for `setexternal` and `setexternal`
- `setexternal` - set external angles for appropriate input type (q.v. section 4.2)
- `setinternal` - set internal angles for appropriate input type (q.v. section 4.2)
- `simparse` - takes in the *simId* string and returns its components in the form usable to the `makeh` function
- `whatmode` - labels the operating mode for `makelist` according to the number of input arguments

11 Features for Future Release

- Implement filter to support arbitrary sample rates for pre-generated waveforms
- Implement additional detectors
- Include data set for galactic population simulations
- Include additional waveforms in `makeh` (suggestions welcome)
- Improve error tracking

A GravEn Testing Suite

Test codes have not been subjected to PSURG code review.

A.1 `testall`

`testall` is the master testing function that will indicate when the results of the code have changed from the results deemed accurate through PSURG code review.

There are no inputs into `testall` and will return the logical gravenPass (1 if all tests ran without error, 0 otherwise). The syntax for `testall` is:

```
gravenPass=testall
```

As the function progresses through the various tests, a pass or fail statement will be output to the monitor as well as written to a file. `testall` generates its own file names to reference the date that the tests were run. The format of the file name is:

```
gravenBuildTest_<date in month_day_year>.txt
```

An example file name is the following:

```
gravenBuildTest_5_12_2004.txt
```

The `testall` function was designed to be run in conjunction with a nightly build routine. Please see the `testall` source code for details on testing criteria.

A.1.1 Example Output from `testall`

The following is an example output file for `testall`:

```
# This GravEn build test was run on 5/12/2004 at 11:  7:31
PASS: graven, original waveform logfile as input file
PASS: graven, pre-generated waveform logfile as input file
PASS: graven, mode 1 (read line-by-line from file)
PASS: graven, mode 2 (make N random draws on file)
PASS: graven, mode 3 (all user input)
PASS: graven, mode 4, ilwd (pre-generated waveform)
PASS: graven, mode 4, txt (pre-generated waveform)
PASS: makelist, mode 1
PASS: makelist, mode 2
PASS: makelist, mode 3, H1
PASS: makelist, mode 3, H2
PASS: makelist, mode 3, L1
PASS: makelist, mode 4, H1
PASS: makelist, mode 4, H2
```

```
PASS: makelist, mode 4, L1
PASS: makelist, mode 5, ampl fixed, internal='optimal', external='random'
PASS: makelist, mode 5, ampl log dist., internal double, external double
PASS: makelist, mode 5, ampl lin.  discrete, internal struct, external struct
PASS: makelist, mode 6, H1
PASS: makelist, mode 6, H2
PASS: makelist, mode 6, L1
PASS: getifo, H1
PASS: getifo, H2
PASS: getifo, L1
PASS: ifodelay, H1
PASS: ifodelay, H2
PASS: ifodelay, L1
PASS: simparse
PASS: makeh, G~000_0.01~0.2
PASS: makeh, SG~512_0.01~0.2
PASS: makeh, CG~512_0.01~0.2
PASS: makeh, BH~512_0.01~0.2
PASS: makeh, SIN~512_000~0.2
PASS: makeh, COS~512_000~0.2
PASS: makeh, SINC~512_000~0.2
PASS: makeh, CHIRP~512_000~0.2
PASS: makeh, SQUARE~512_000~0.2
PASS: makeh, SQUAREG~512_0.01~0.2
PASS: makeh, SAWTOOTH~512_000~0.2
PASS: makeh, SAWTOOTHG~512_0.01~0.2
PASS: makett
PASS: detproj, H1
PASS: detproj, H2
PASS: detproj, L1
# GravEn build test ran successfully without error!
# +++
```

A.2 `testdetproj`

`testdetproj` yields a graphical representation of the antenna pattern of a detector as well as being used by `testall` to compare current version results with its reference data (q.v. appendix B).

The antenna pattern is defined to be:

$$\rho = F_+^2 + F_\times^2 \tag{23}$$

Example data for the antenna pattern for LHO is in figure 4 and the antenna pattern for LLO is in figure 5.

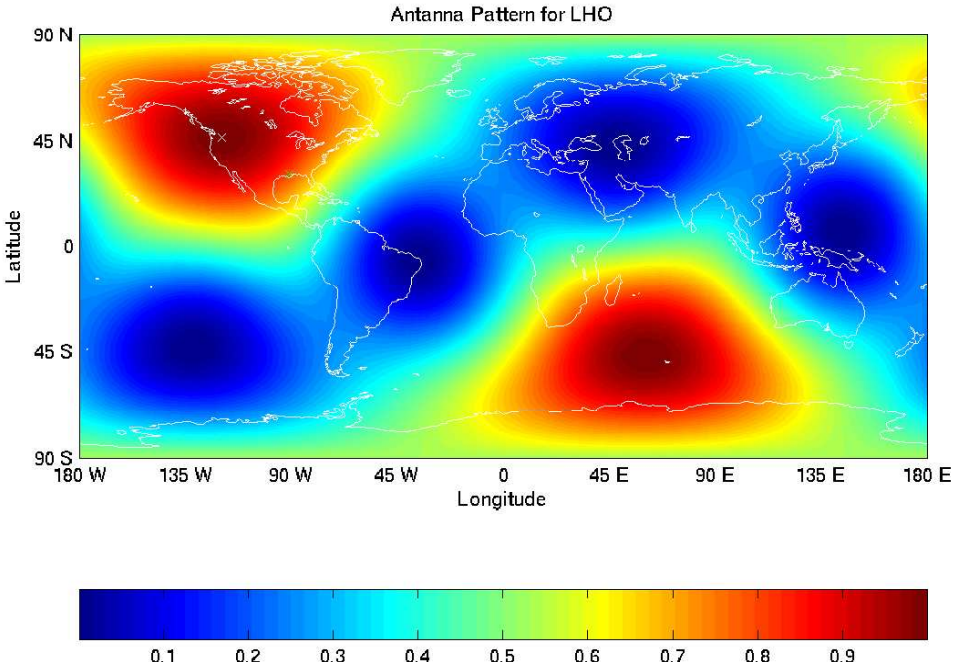

Figure 4: Antenna pattern for LHO

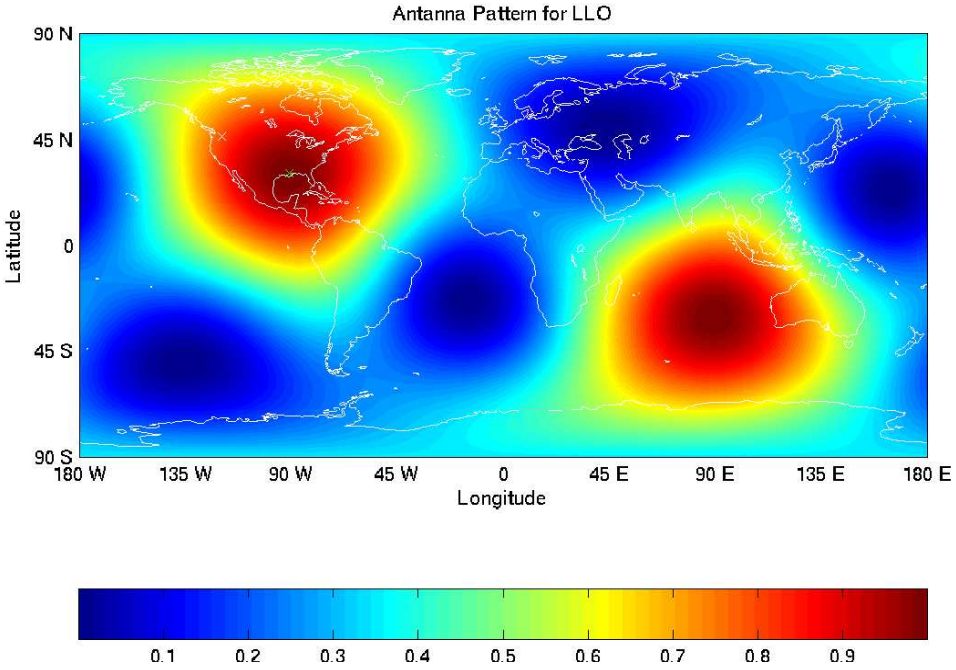

Figure 5: Antenna pattern for LLO

The syntax for `testdetproj` is:

```
rho=testdetproj(detId,plotSwitch)
```

where `plotSwitch` is the optional string `'off'` and is used to suppress the graphics display (this is especially useful for using `testdetproj` inside of `testall`). `rho` is a 101×101 element array that samples the sky between 0 an π in increments of $\pi/100$ along the co-declination and between -π and π in increments of $\pi/50$ along the longitude (centered on the Prime Meridian).

A.3 `testgetifo`

`testgetifo` compares the coordinates output by `getifo` to those deemed correct in its reference data set (q.v. appendix B).

The syntax for `testgetifo` is:

```
[errorH1,errorH2,errorL1]=testgetifo
```

where `error<detId>` is a 1×7 array of logicals indicating if the results of the current version are different from the reference version. If an element is true (i.e. 1) then there is a change from the reference results. Each element of the logical array is:

```
[V, W, LAT, CODEC, LONG, WGS84, R]
```

where `V` is the unit vector for the X-arm, `W` is the unit vector for the Y-arm, `LAT` is the latitude of the beam splitter, `CODEC` is the cosine of the co-declination of the beam splitter, `LONG` is the longitude of the beam splitter, `WGS84` is the WGS-84 coordinates of the beam splitter in meters and `R` is the distance of the beam splitter from the center of the Earth in meters.

These logicals are then interpreted by `testall` to generate the pass/fail message.

A.4 `testifodelay`

`testifodelay` yields a graphical representation of the number of samples to add to a start time to account for the time of flight from the center of the Earth to the detector given the source's sky location and is used by `testall` to compare current version results with it reference data (q.v. appendix B).

Example data for LHO is shown in figure 6 and LLO in figure 7.

The syntax for `testifodelay` is as follows:

```
delay=testifodelay(detId,sampFreq,plotSwitch)
```

where `plotSwitch` is the optional string `'off'` and is used to suppress the graphics display (this is especially useful for using `testifodelay` inside of `testall`). `delay` is a 101×101 element array that samples the sky between 0 an π in increments of $\pi/100$ along the co-declination and between -π and π in increments of $\pi/50$ along the longitude (centered on the Prime Meridian).

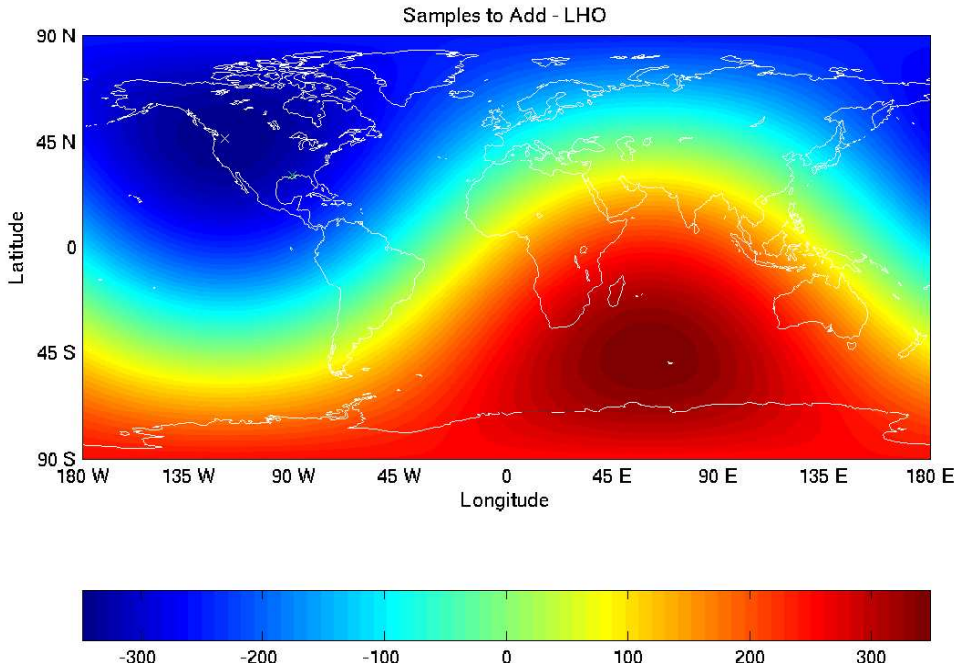

Figure 6: The number of samples to add to signal start time with respect to the source sky location for LHO

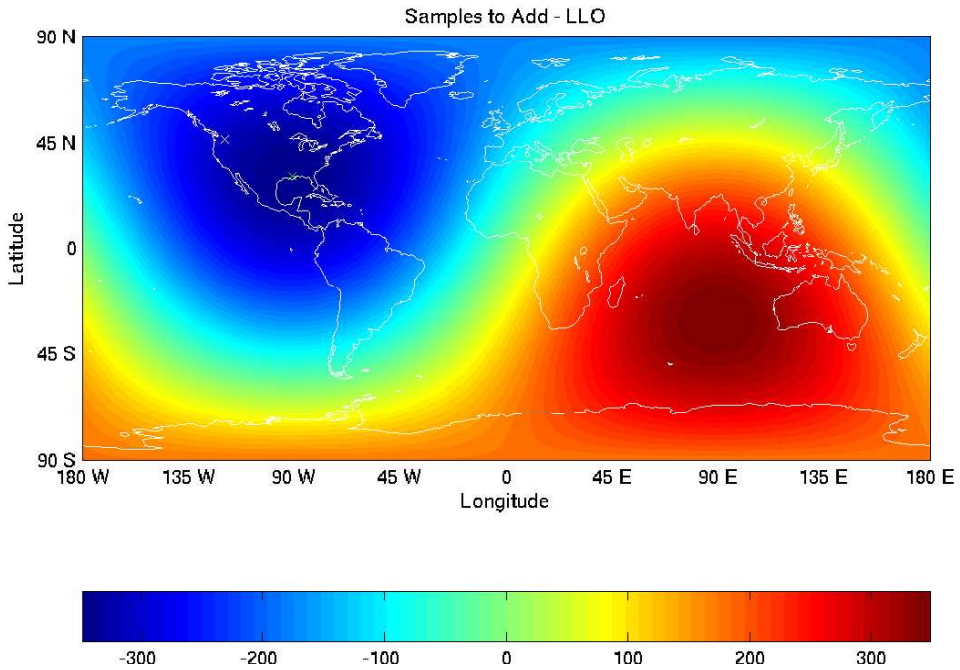

Figure 7: The number of samples to add to signal start time with respect to the source sky location for LLO

Given the WGS-84 coordinates for the beam splitter (from `getifo`) the maximum number of samples to add for LHO is approximately 347.9687 samples and the maximum number of samples to add for LLO is approximately 348.2718 samples.

A.5 `testmakeh`

`testmakeh` is a visualization utility and not used by `testall`. Since the output of `makeh` is in 3×3×time format, it is difficult to inspect the time evolution of a single element. `testmakeh` allows a user to specify a *simId* and view the time evolution of all 9 elements (q.v. figure 8) or specify a *simId* and element row and column to view the time evolution of a single element (q.v. figure 9).

In figure 8, the row number increases from top to bottom and the column number increases left to right.

The syntax for `testmakeh` is:

```
[timeSeries,hij]=testmakeh(simId)
```

to view all 9 components or:

```
[timeSeries,hij]=testmakeh(simId,row,column)
```

to view the (row,column) component. `timeSeries` is the time evolution of the selected component(s) and `hij` is the metric perturbation in full 3×3×time format.

A.6 `testmakett`

`testmakett` tests `makett`'s results for projecting metric perturbations into the TT gauge against results deemed to be correct in its reference data set (q.v. appendix B).

The syntax for `testmakett` is:

```
errorMakeTt=testmakett
```

where `errorMakeTt` is a 1×3 array of logicals indicating if the results of the current versions are different from the reference version. If an element if true (i.e. 1) then there is a change from the reference results. Each element in the logical array is:

```
[hijX, hijY, hijZ]
```

where `hijX`, `hijY` and `hijZ` are metric perturbations whose gravitational waves are propagating in the X, Y and Z directions, respectively.

These logicals are then interpreted by `testall` to generate the pass/fail message.

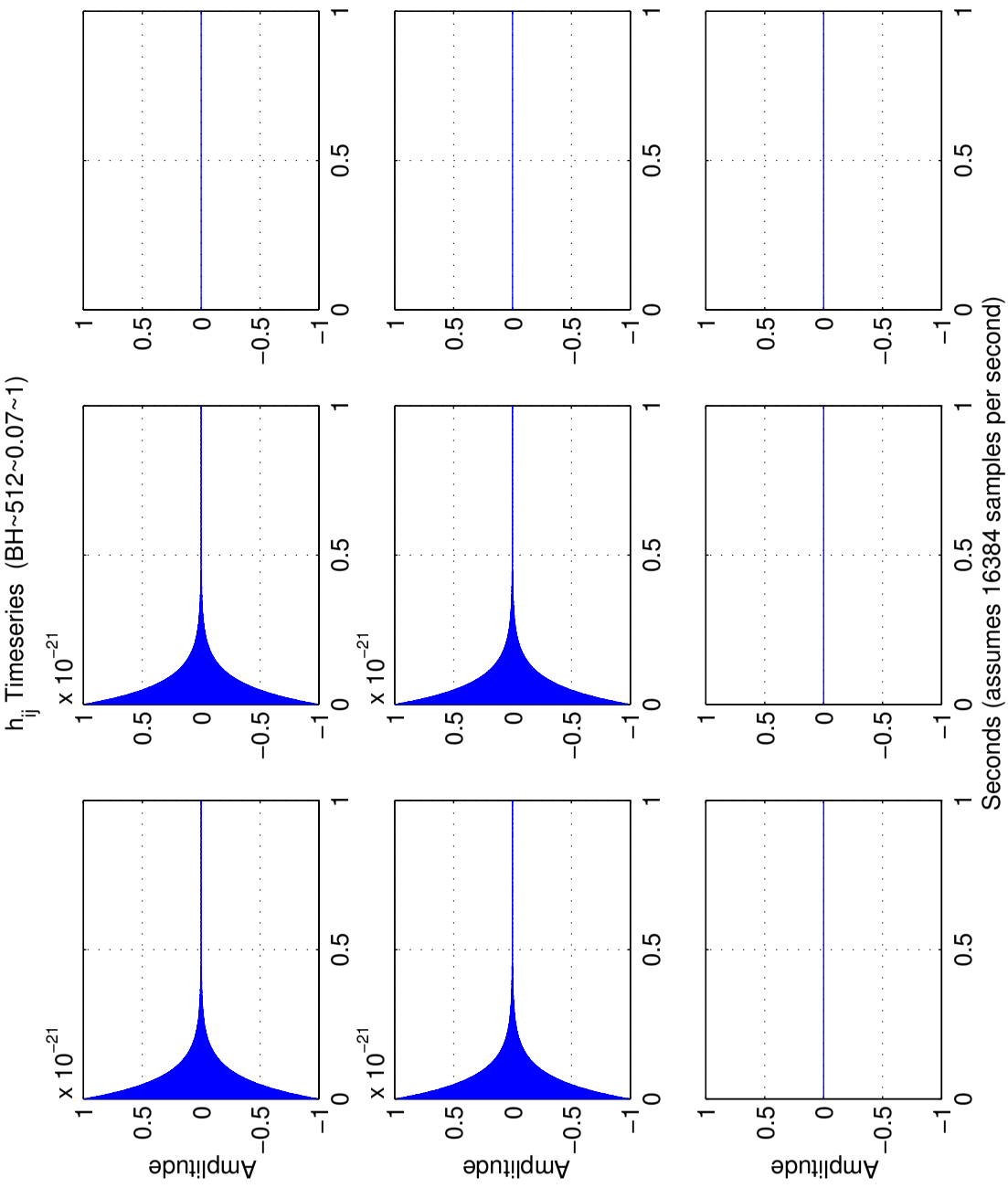

Figure 8: Graphical representation of each element timeseries in h_{ij}

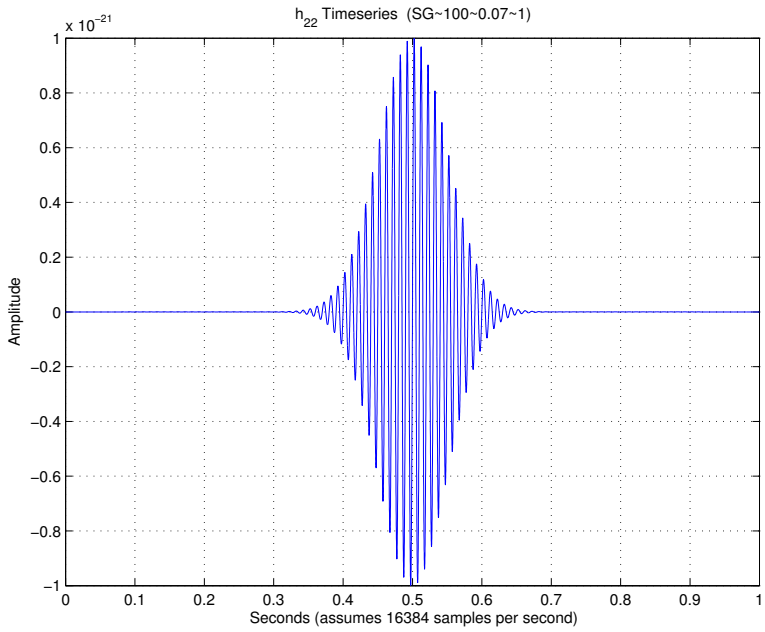

Figure 9: Graphical representation of the (2,2) element timeseries in h_{ij}

A.7 `testsimparse`

`testsimparse` tests `simparse`'s results for parsing the *simId* string `'test~0123_4.56~7.89'` against results deemed to be correct in its reference data set (q.v. appendix B).

The syntax for `testsimparse` is:

```
errorSimparse=testsimparse
```

where `errorSimparse` is a 1×4 array of logicals indicating if the results of the current version were different from the reference version. If an element is true (i.e. 1) then there was a change from the reference results. Each element in the logical array is (q.v. section 5.2):

```
[id, f, tau, dur]
```

These logicals are then interpreted by `testall` to generate the pass/fail message.

A.8 `ligomap` Utility

`ligomap` is a utility that plots a two-dimensional array onto a map of the world with the locations of LHO and LLO marked. It produces both 3-dimensional and 2-dimensional plots.

The syntax for `ligomap` is:

```
[Coast,Llo,Lho]=ligomap(xx,yy,zz)
```

where `xx` is a vector defining the row sampling of the array, `yy` is a vector defining the col-

umn sampling of the array, `zz` is the array to be plotted on world map (N.B. must be `xx`×`yy` in size), and `Coast`, `Llo` and `Lho` are the coordinates of the coastlines, LLO and LHO that where plotted. The inputs for `ligomap` are much the same as for the MATLAB built-in functions `mesh` and `surf`.

Figures 4, 5, 6 and 7 were generated using `ligomap`.

B Reference Data Sets

In order to compare results of the current version of GravEn to the results of previous versions that have been deemed correct, it is useful to have a library of reference data sets. Unless otherwise stated, all of these reference data sets are used by the `testall`. Also, all of the reference data sets used by `testall` follow the naming convention:

```
<function being tested>Data<data generated in dayMoYear>.mat
```

In the event that newer reference data sets are available, please use the most current data.

Reference data sets are assumed to be in a child folder named `refData` within the folder where the test suite is saved.

- `coast.mat` - data set used by `ligomap` to plot the coastlines of the world map (not used by `testall`)
- `detprojData21Apr2004.mat`
- `getifoData5May2004.mat`
- `gravenData7May2004.mat`
- `ifodelayData21Apr2004.mat`
- `ligomapdata.mat` - alternate data set to `coast.mat` if the `ligomap` is not present (not used by `testall`)
- `makehData5May2004.mat`
- `makelistData6May2004.mat`
- `makettData5May2004.mat`
- `pregentest.txt` - the pre-generated waveform contained in `ZM_A1B1G1.ilwd` used to test txt-file reading abilities for pre-generated waveforms
- `simparseData5May2004.mat`
- `testopt.txt` - test input file that contains angles that are specified to be both random and optimal
- `testrand.txt` - same as `testopt.txt` except angles are not specified as optimal

- `ZM_A1B1G1.ilwd` - pre-generated waveform used to test the ilwd-file reading abilities for pre-generated waveforms; this is the A1B1G1 waveform from the Zwerger-Muller supernova waveform catalog sampled at 16384 samples per second found at:

http://www.ldas.ligo-wa.caltech.edu/ldas_outgoing/jobs/ilwd/burstsim/