LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY

- LIGO -

CALIFORNIA INSTITUTE OF TECHNOLOGY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	**LIGO-T040020-00-Z**	2004/03/23
GravEn Simulation Engine Primer		
Amber L. Stuver		

Distribution of this draft:

Draft: Circulation restricted to LIGO-I members

PRELIMINARY DOCUMENTATION PENDING PSURG CODE REVIEW

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 2004/03/23

Contents

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.

This primer is **preliminary** pending PSURG code review and public release[1]. For further information, 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 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 measured clockwise along Earth's longitude (in rad)
- *fname* - (string) name of file containing information for generating simulations
- *fs* - sampling frequency of detector (in Hz)
- *GPS* - GPS start time (in whole seconds)
- *internal* - structure containing description of source's orientation to line of sight
  - *.x* - cosine of the angle the line of sight makes with the source's axis of angular momentum

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

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

- *logfname* - (string) name of log file
- *N* - number of simulations to generate
- *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)**
- *stime* - 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:

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

  ```
  graven(logfname,GPS,detID,fname,fs,seed)
  ```

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

  ```
  graven(N,logfname,GPS,detID,fname,fs,seed)
  ```

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

  ```
  graven(N,logfname,GPS,detID,simID,ampl,stime,internal,external,fs,seed)
  ```

3.2 Structure of the Driver

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. The loop then increments and goes through the next line of the master simulations list.

3.3 Outputs

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

- *sim* - vector of structures containing the simulations
  - *.nstr* - the start time of the simulation in whole samples
  - *.sig* - the simulation time series in units of AS_Q counts
  - *.gps0* - the start time of the simulation in seconds and nanoseconds
  - *.PKgps* - the time of maximum strain in seconds and nanoseconds

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(fname)`

- 2 inputs - generate simulations list from *N* random draws on the lines of a file

  `makeList(N,fname)`

- 6 input - generate simulations list from inputs directly

  `makeList(N,simID,ampl,stime,internal,external,detID` *`(optional)`*`)`

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'`[2]. In this case, the needed angles are chosen from the appropriate ranges.

  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 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 have the following format:

```
simID ampl stime1 stime2 int1 int2 ext1 ext2 ext3
```

where the range of allowed start times (in samples) is given by `stime1` to `stime2` (lower to upper), internal angles *internal.x* and *internal.phi* are `int1`, and `int2`, and the external angles

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

external.x, *external.phi* and *external.psi* are `ext1`, `ext2`, and `ext3`.

Sample file:

```
SG_512_0.01_0.25 1e-21 10 20 -0.707 +0.7941 +0.426 +0.349 -1.6755
G_525_0.015_0.2 1e-21 100 200 RANDOM Random random RANDOM RANDOM
G_475_0.02_0.4 1e-33 1024 16384 Random +0.6590 random -1.7453 RANDOM
CG_530_0.01_0.2 1.25e-20 1024 4096 0.001 -2.1817 +0.5001 1.0472 1.1345
```

4.4 Outputs

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

```
simID ampl stime int1 int2 ext1 ext2 ext3
```

where `stime` 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)
- *fs*

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. The format for this is:

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

e.g. `'SG_530_0.01_0.25'` 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[3].

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

[3]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.

5.3 Current Supported Waveforms

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

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

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

$$CG(f,t,\tau) = ampl * cos(2\pi ft) * e^{-(\frac{t}{\tau})^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3}$$

$$BH(f,t,\tau) = ampl * e^{-\frac{t}{\tau}} \begin{pmatrix} sin(2\pi ft) & cos(2\pi ft) & 0 \\ cos(2\pi ft) & sin(2\pi ft) & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4}$$

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).

The source's $\hat{Z}$ axis is rotated to coincide with $\hat{n}$:

$$\hat{n} = sin(\theta)cos(\varphi)\hat{x} + sin(\theta)sin(\varphi)\hat{y} + cos(\theta)\hat{z} \tag{5}$$

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

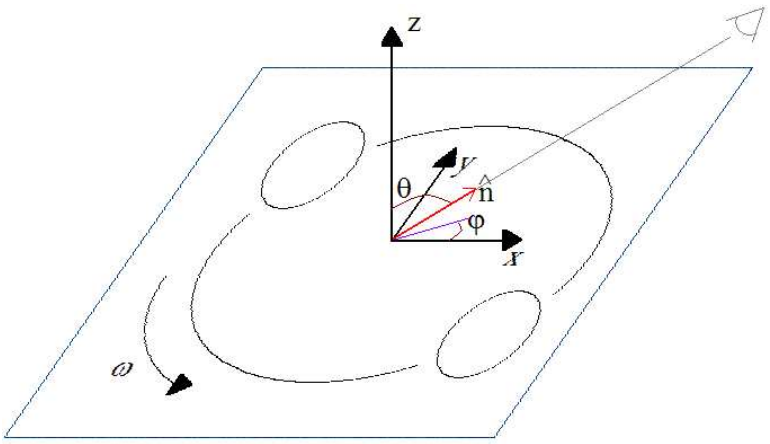

Figure 1: Illustraion of source internal angles

From the definition of $\hat{n}$, the projection operator for the TT gauge is constructed:

$$P_{ij} = \delta_{ij} - n_i n_j \tag{6}$$

Using equation 4 of box 35.1 in MTW[4], the metric perturbation is put into the TT gauge by:

$$h_{ij}^{TT} = P_{il} h_{lm} P mj - \frac{1}{2} P_{ij} (P_{lm} h_{lm}) \tag{7}$$

From the form of the TT gauge for a gravitational wave propagating in the $\hat{n}$ direction (along the line of sight):

$$h_{ij}^{TT} = \begin{pmatrix} h_+ & h_\times & 0 \\ h_\times & -h_+ & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{8}$$

it is seen that only two components need to be computed to fully describe the gravitational wave. Only the + polarization component h_{11}^{TT}, and the $\times$ polarization component h_{12}^{TT} are calculated.

6.3 Outputs

- The `2×time` vector containing the + and $\times$ polarizations:

  (1,:) - + polarization

  (2,:) - $\times$ polarization

7 Detector Projector - `detProj`

7.1 Inputs

- The `2×time` vector containing the plus and cross polarizations of the gravitational wave

[4]Misner, C.W., Thorne, K.S., Wheeler, J.A. *Gravitation*. San Francisco: W.H. Freeman, 1973

- *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 incident to the North Pole. In order to project a gravitational wave to an arbitrary sky location, the gravitational wave is rotated off of the North Pole to the source's sky location:

$$R = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ sin(\theta)sin(\varphi) & cos(\varphi) & -cos(\theta)sin(\varphi) \\ sin(\theta)cos(\varphi) & sin(\varphi) & cos(\theta)cos(\varphi) \end{pmatrix} \tag{9}$$

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.

This rotation is then applied to the TT gauge form of the gravitational wave (by definition, the gravitational wave's propagation is in the $\hat{Z}$ direction):

$$h_{ij}^{TT\prime} = R^{-1} \begin{pmatrix} h_+ cos(2\psi) & h_\times sin(2\psi) & 0 \\ h_\times sin(2\psi) & -h_+ cos(2\psi) & 0 \\ 0 & 0 & 0 \end{pmatrix} R \tag{10}$$

where ψ is the polarization angle *external.psi*. ψ is the gravitational wave's rotation in the plane perpendicular to the direction of propagation measured clockwise from $\hat{X}$.

The antenna pattern is formed by:

$$D = \hat{V} \otimes \hat{V} - \hat{W} \otimes \hat{W} \tag{11}$$

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.

The final detector projection is the contraction of the antenna pattern and the projection of the gravitational wave:

$$h = -\frac{1}{2} h_{ij}^{TT\prime} D^{ij} \tag{12}$$

7.3 Outputs

- The final result of this detector projection is the simulated waveform timeseries in units of strain

8 Inverse Calibrator - `calibsim_fd`

8.1 Inputs

- *detID*
- *GPS*
- The simulation timeseries in units of strain
- *fs*
- 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:

$$R(f,t) = \frac{1 + \alpha(t)\beta(t)H(f)}{\alpha(t)C(f)} \tag{13}$$

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[5].

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

$$\frac{1 + \alpha(t)\beta(t)H(f)}{\alpha(t)C(f)} = \frac{1}{\alpha(t)C(f)} + \frac{\beta(t)H(f)}{C(f)} \tag{14}$$

The inverse calibration cannot be separated so easily:

$$\frac{\alpha(t)C(f)}{1 + \alpha(t)\beta(t)H(f)} \neq \frac{\alpha(t)C(f)}{1} + \frac{C(f)}{\beta(t)H(f)} \tag{15}$$

This property has created difficulties in producing the inverse calibration in the time domain. However, the frequency domain calibration has proved 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:

$$R^{-1}(f,t) = \frac{\alpha(t)C(f)}{1 + \alpha(t)\beta(t)H(f)} \tag{16}$$

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

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

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) and sensing function (C) file

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

```
frequency magnitude phase
```

so that

$$H(f)\ \&\ C(f) = magnitude(f) * e^{i*phase(f)} \tag{17}$$

Sample open loop gain (H) or sensing function (C) 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`

8.5 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 files 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 stime1 stime2 internal.x internal.phi ...
            external.x external.phi external.psi refGPS ...
                    sigGPSsec sigGPSns 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, `stime1` must equal `stime2` and are in units of samples with respect to `refGPS` (`refGPS`=*GPS*), which is the simulation start time at the center of the Earth. `sigGPSsec` and `sigGPSns` 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.

10 Other Needed Functions

- `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)
- `simParse` - takes in the *simID* string and returns its components in the form usable to the `makeh` function
- `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 for public release)
- `read_alphabetafile` - reads the $\alpha\beta$ file and returns α and β corresponding to the GPS time of the simulation
- `h2asq_fd` - performs the inverse calibration on the simulated strain data as described in section 8.3
- `ndx2gps` - converts samples to time (in whole seconds and nanoseconds) after the reference *GPS* time in seconds