LIGO-T010099-00-R

GWCLASS Manual

Version 1.0.3.1

J Marsano[1][2], K Riles
Department of Physics
University of Michigan
500 E University Ave
Ann Arbor, MI 48109-1120

September 13, 2001

[1]Present address: Jefferson Physical Laboratory, Harvard University, Cambridge, MA 02138, marsano@fas.harvard.edu
[2]Technical questions concerning GWCLASS code should be addressed to J Marsano

Contents

Chapter 1

Introduction

The purpose of the GWCLASS package is to facilitate the analysis of data taken from gravitational wave interferometers, such as those being constructed in the United States by the LIGO project. The two primary components of this package are a preprocessing function intended to reduce the signal to noise ratio in the primary ADC channel and a search function geared specifically to seek for evidence of periodic gravitational waves emitted by known pulsars. In addition, several auxiliary functions that the author found useful while analyzing gravitational wave data have been included in the distribution. A full listing of the GWCLASS functions is presented in table 1.1. Also included with the distribution are utility scripts, written primarily in Perl, to aid in data analysis, and sample files needed by some of the data processing functions. These will be discussed when needed.

The remainder of this chapter discusses software requirements and the installation procedure. Following this, chapter 2 describes the GWCLASS binaries and how they are used. In addition, the procedures used by `clean_signal` to estimate and remove correlations and `gwsearch` to search for evidence of periodic gravitational waves are described in detail. Finally, a brief description of some of the utility scripts provided with the GWCLASS distribution is contained within chapter 3.

Finally, the GWCLASS source code may be obtained at either the GWCLASS homepage or the mirror site, whose addresses are listed below.

GWCLASS Homepage	`http://www.fas.harvard.edu/~marsano/GWCLASS`
Mirror Site	`http://www-mhp.physics.lsa.umich.edu/~keithr/GWCLASS`

1.1 Hardware and Software Requirements

GWCLASS was developed primarily on a PC with an Intel Pentium IV Processor running RedHat Linux version 7.0. It has been successfully compiled on a number of other architectures, though, including SGI Irix, HPUX, and SPARC. Since nothing in the code is architecture-specific, it is not anticipated that any hardware incompatibilities exist provided the proper compiler flags are issued.

On the other hand, there are a number of software components that are needed by GWCLASS. Software which is required for the compilation of GWCLASS binaries, as well as the corresponding websites at which the software may be obtained, are listed in table 1.2. Additional software that must be used in conjunction with GWCLASS in order to enable some of its features, along with corresponding websites websites, are listed in table 1.3.

Function	Description
clean_signal	Estimates and removes correlations between the signal ADC and environmental ADC's Outputs new Frames containing a "cleaned" signal ADC
dump_frame	Dumps all data for a given channel from a specified Frame to an ASCII text file
gps2gmst	Converts a specified time in GPS seconds to GMST (Greenwich Mean Sideral Time)
gps2mjd	Converts a specified time in GPS seconds to MJD (Modified Julian Date)
gwsearch	Performs a search algorithm to detect periodic gravitational waves from a known pulsar If a detection occurs, the magnitude, h, and its uncertainty are printed
lockcat	Produces a catalog of locked sections within a user-provided list of frames
plot_frame	Interface to xmgrace which plots the fourier transform of Frame data
print_freq	Prints frequency of a pulsar during a specified time interval to an ASCII text file
read_data	Prints a specified amount of data from a given channel and prints it to an ASCII file Similar to dump_frame except it may print data obtained from a series of Frames or from only a fraction of a Frame
utc2mjd	Converts a specified UTC time to MJD (Modified Julian Date)

Table 1.1: Functions included in GWCLASS distribution

In addition to using the software listed in table 1.2, GWCLASS also makes use of code from the GRASP analysis package written by Bruce Allen and others[1] [1]. In particular, three source files from the GRASP package, `frameinterface.c`, `utctime.c`, and `GR_error.c`, have been integrated into GWCLASS and are included in the GWCLASS distribution. The first, `frameinterface.c`, contains functions for reading data from Frame files and has been edited in order to permit GWCLASS functions to use FTP or HSI (an interface to High Performance Storage Systems or HPSS) to read a continuous stream of data stored on a remote server. The second, `utctime.c`, is used to incorporate leap seconds into the conversion from calendar time to UTC (and GPS) seconds. Finally, `GR_error.c` is included to print status and error messages produced by `frameinterface.c` and `utctime.t` in the GRASP format as a recognition that GRASP code is being used. Due to the fact that at least one of the GRASP source files have been edited for inclusion into GWCLASS, all three are included in the GWCLASS distribution. To be sure, a pre-existing GRASP installation is not necessary, and if present is not referenced.

1.2 Installation

After the necessary software has been installed, the GWCLASS binaries may be compiled by first editing `GWCLASS_dir/src/Makefile`, where `GWCLASS_dir` is the top directory of the GWCLASS installation, and running `make` from `GWCLASS_dir/src`. Individual binaries may be compiled by issuing `make <name>`, where `<name>` is the name of the desired binary. The instructions for editing `Makefile` are contained therein.

[1] The latest version of GRASP may be obtained at `http://www.lsc-group.phys.uwm.edu/ ballen/grasp-distribution`. Note, however, that GWCLASS will not be compatible with any distribution of GRASP. To be sure, GWCLASS must be compiled with the copy of `frameinterface.c` provided with the GWCLASS distribution.

Software Package	Location/Description
CLAPACK	`http://www.netlib.org/clapack` Linear algebra package required to solve matrix equations and find eigenvalues/vectors
FFTW	`http://www.fftw.org` Package for compute FFT's (Fast Fourier Transforms)
Frame Library	`http://wwwlapp.in2p3.fr/virgo/FrameL` Required to read interferometer data from Frames **WARNING:** GWCLASS has only been tested with Frame library version 3.85. It may not be compatible with future versions at the present time
Grace^2	Graphics package and successor to Xmgr

Table 1.2: Required software for GWCLASS installation

Software Package	Location/Description
hsi	`http://www.sdsc.edu/Storage/hsi` Interface to High Performance Storage System (HPSS) If present, GWCLASS may use hsi to obtain and/or store Frame files on a remote HPSS server
ncftp version $\geq$ 3.0	`http://www.ncftp.com` File Transfer Protocol (FTP) function GWCLASS needs the functions `ncftpput` and `ncftpget`, which are included in the ncftp distribution versions 3.0 and greater, in order to use FTP to obtain and/or store Frame files on remote servers
tempo	`http://pulsar.princeton.edu/tempo` Pulsar timing code used for many pulsar searches in the astronomical community GWCLASS uses output from tempo to determine the frequency modulation of known pulsars. In order to use the search function, `gwsearch`, or signal injection feature of the cleaning function, `clean_signal`, the user must provide the proper tempo output files.
gzip	`http://www.gzip.org` Required for storing Frames produced by `clean_signal` on a remote server. The Frames are automatically compressed with gzip before being transferred

Table 1.3: Software required to use certain GWCLASS features

Chapter 2

GWCLASS binaries

The crux of GWCLASS are the cleaning and search binaries, `clean_signal` and `gwsearch`. As a result of this, the bulk of this section describes their usage. The other utility functions included with GWCLASS will then discussed briefly.

2.1 Function: `clean_signal`

The `clean_signal` binary reads data from Frame files, estimates correlations between the primary signal channel and environmental monitor channels, hereafter referred to simply as environmental channels, and removes them in order to improve the signal to noise ratio. The procedure by which this is accomplished is a very slight modification of that described in a paper by Allen, Hua, and Ottewill[2] For completeness, the method is described in full in section 2.1.3. Following the removal of correlations, `clean_signal` creates new frames containing the initial signal channel, the signal channel following removal of environmental correlations, and the environmental channels whose correlations were removed. The details concerning input and output are described below.

2.1.1 Command Line Usage

The user interacts with `clean_signal` in two ways, namely by using command line options and by setting environment variables. The command line options control virtually all of the functioning of `clean_signal`, with the environment variables serving only to configure the method by which Frame files are obtained and stored. The command line usage for `clean_signal` reads as follows:

```
Usage: clean_signal [options] [configuration file]

Configuration file should contain, in columns, the names of the channels
to be sent to the correlation algorithm, with the signal channel first.

Options:
   -b<integer>   Number of frequency points in each frequency band
                    used by correlate_chans function
                    (DEFAULT = 128)
   -c<real>      Pairwise covariance coefficient threshold below which
                    correlation is assumed to be statistically
                    insignificant
                    (DEFAULT = 10/band_length, band_length given by
                   -b option
-e<real>      Threshold used to determine when an eigenvalue of the
                 correlation matrix is sufficiently near zero that
                 the correlation matrix must be reduced
                 (DEFAULT = 1e-5)
-f<integer>   Sets number of times to apply notch filter to remove
                 line harmonics to <integer>
                 (DEFAULT = 0)
-F<real>      Sets critical frequency (in Hz) whose harmonics are to be
                 removed by the notch filter to <real>
                 (DEFAULT = 60)
-h            Help (prints this list)
-H            Prints help information which details the environment varaibles
                 that must be set when using clean_signal
-i<string>    Inject false signals into the data using the configuration
                 file indicated by <string>.  That file has one line for
                 each pulsar for which a false signal is injected, which
                 has the following format:

                 PLUS <plus amplitude> <plus phase> CROSS <cross amplitude> <cross phase>
                 Frequency information will be obtained from the file
                 <pulsar name>_polyco.dat, which is a polyco file produced
                 by tempo with the -z switch.  Right ascension and declination
                 will be obtained from the file <pulsar name>.param, which
                 should contain these items, in order, on the first line
                 expressed in RADIANS
-n<integer>   Sets number of data points to obtain from the signal
                 channel and clean at once to 2^<integer>
                 (DEFAULT = 17)
-s<integer>   Sets preferred length of cleaned segments of data points
                 printed to FRAMES to 2^<integer>
-v            Verbose
-w<string>    Sets window function to be used when computing FFT to
              <string>.  Supported window functions are:
              NONE
              HANNING (This is the default window if none is indicated)
```

We now embark upon a more detailed description of these options.

Correlation Parameters

The `-b`, `-c`, and `-e` options configure parameters used when estimating correlations and will be explained when the procedure by which this is accomplished is discussed below. Generally, the defaults for these options work fine.

Notch Filters

The `-f` and `-F` options toggle the use of notch filters prior to correlation estimation. The inclusion of this feature was motivated by the desire to remove the 60 Hz harmonics from the data prior to processing. When the `-f` option is set, notch filters are applied to remove lines at the frequency indicated by the `-F` option (the default is 60 Hz) as well as all of the harmonics of that frequency contained within the FFT's frequency range. Note that it is **NOT** currently possible to remove a single line without removing its harmonics. The integer accompanying the

`-f` option determines the number of times each filter is applied and may be used to increase the effectiveness of line removal if one application doesn't do the trick. Note that an integer **MUST** be given with the `-f` option, even if the filter is only to be applied once (in that case, the option given should be `-f1`).

Injection of false signals

`clean_signal` has the ability to inject artificial signals, corresponding to periodic gravitational waves from given pulsars, that exhibit both the appropriate frequency and amplitude modulation. To do this, `clean_signal` requires a number of input files. The file indicated with the `-i` option should contain names of the pulsars for which signals are to be injected and the amplitudes and phases of the plus and cross polarization comopnents of those signals. The format of this file is described in the usage above. The phases are relative to the first time stamp appearing in the first "cleaned" Frame output by `clean_signal`. The plus and cross polarizations are defined in section 2.2.3.

For each pulsar, `clean_signal` requires two additional files, one for the frequency modulation, and the other for the amplitude modulation. Both filenames are determined by the pulsar's name as indicated in the `-i` file. For the determination of frequency modulation, GWCLASS relies on an output file from the TEMPO timing code, used for many pulsar searches in the astronomical community. When called with the `-z` switch, TEMPO produces a file, generally called "polyco.dat", which contains polynomial coefficients that may be used to approximate a pulsar's frequency modulation. `clean_signal` looks for this file under the name <pulsar name>_polyco.dat, where <pulsar name> is the name provided in the `-i` file. Thus, in order to use this feature of `clean_signal`, the user must first run TEMPO in order to generate the appropriate polynomial coefficient file and change its name appropriately. This process is described in greater detail in section 2.2.3, where frequency modulation is discussed in relation to the `gwsearch` algorithm.

In addition to frequency modulation, `clean_signal` also determines the amplitude modulation due to the continuously changing relative orientation of the pulsar and detector. For this determination, `clean_signal` uses the right ascension and declination of the pulsar, which are provided in the file <pulsar name>.param. This is a user-created file which contains two entries, the pulsar right ascension and declination (in that order), on the first line. As a reminder, both the right ascension and declination **MUST** be in **RADIANS**. The process by which amplitude modulation is determined is described in greater detail in section 2.2.3, where it is discussed in relation to the `gwsearch` algorithm.

Setting the Amount of Data

The data to be processed is stored in Frames either in a local directory or on a remote server, with the details indicated by environment variables. This is discussed below. While `clean_signal` will always continue running until it exhausts all of this data, the size of the data chunks which are read and processed is set with the `-n` and `-s` parameters. To understand exactly what these parameters do, it is necessary to digress slightly and describe briefly some of the workings of `clean_signal`.

When called, `clean_signal` will attempt to obtain 2^x data points, where x is the integer provided by the `-n` switch, from the signal channel and a number of data points corresponding to an equivalent amount of time from each environmental channel. This data is then printed to a series of buffer files, at which point `clean_signal` reads another chunk. Eventually, however, either two successive chunks are read which are not adjacent in time (because the interferometer fell out of lock sometime between the end of the first and beginning of the second), or it is determined that a critical number of data points, namely 2^y where y is the integer provided by the `-s` switch, have already been written to the buffers. At that time, `clean_signal` proceeds to process the data already contained in the buffers before dealing with the most recent chunk.

The result is that `clean_signal` will produce Frame files of length corresponding to the `-s` switch whenever possible (ie when breaks in lock don't interfere) and will produce Frame files of smaller length but no smaller than that corresponding to the `-n` switch otherwise. More importantly, though, `clean_signal` will never attempt to read more than 2^x data points at a time. This is critical since it is often desirable to simultaneously process a length of data that is so large that there is not enough data to read all of it at once. So, `-n` can be set sufficiently small that memory problems are avoided while `-s` can be set larger since memory problems are avoided in the processing phase through the use of buffers. It may still be possible to choose a value of `-s` and a number of data channels that exhausts the memory on a given system. When this happens, a malloc error will be printed.

Help and Verbose

There are two help options, `-h` and `-H`. The former prints the command line usage above to screen, while the latter prints a help file that describes the environment variables that must be set in order to control the input and output of `clean_signal`. These environment variables will be discussed below. The use of `-v` toggles verbose mode, in which additional messages describing the activities and progress of `clean_signal` are printed to screen.

Miscellaneous

Two miscellaneous options remain, namely `-m` and `-w`. The `-m` option specifies the maximum frequency, in Hertz, at which cleaning occurs. The default is set to 1024 Hz since most astrophysical sources will be found at frequencies less than this. The reason for the inclusion and use of this option is to save memory and computation time. Perhaps another digression will be useful here. When `clean_signal` begins processing data that has been written to buffers, data from the buffers is read one channel at a time, with the signal channel read last, in order to save memory space. After being read, each channel is FFT'ed, at which point only those FFT coefficients corresponding to frequencies less than or equal to that indicated with the `-m` option are kept. Since the signal channel is read last, though, its entire FFT is kept to enable later reconstruction of the signal (the FFT is stored in the array used previously to read from buffers). When all channels are read, the correlation algorithm described below in section 2.1.3 is applied, and the FFT coefficients of the signal channel in the working frequency range are modified. Finally, an inverse FFT is applied to obtain the "cleaned" signal. From this discussion, it is clear that the value given by the `-m` option determines the number of FFT coefficients that must be stored at once for the correlation algorithm to proceed. If the system's memory becomes exhausted, this parameter may be decreased in order to help alleviate the problem.

The `-w` option indicates the window function that is to be applied to channel data prior to being Fourier transformed. The only supported options at this time are NONE, which is equivalent to a rectangular window, and HANNING. All windowing is done through the window() function located in window.c so, to add additional windows, only this file need be edited.

2.1.2 Environment Variables

While command line arguments control most of the operation of `clean_signal`, input and output are controlled through the use of a number of environment variables. These variables retain the prefix "GRASP_" in deference to the fact that the functions used to obtain data from Frames were originally part of the GRASP distribution. Only the GRASP_FRAMEPATH variable was used by GRASP, though. The rest have been added in order to permit input from and output to remote servers via FTP and HSI. What follows is the `-H` help for environment variables.

```
*** Environment Variables ***

The function clean_signal utilizes a frame interface that uses
several environment variables documented below:

* Input Frames:

    There are two methods of providing input frames for clean_signal
    One method is to indicate a local directory in which
    input frames are located.  The other is to indicate a
    remote host on which tar archives containing input frames
    are located, the remote directory in which they are located,
    a list of tar archives to obtain, and a local directory to
    which clean_signal transfers them before reading
    This latter method requires either an hsi interface to tar
    archives stored on an hpss server or a local installation of
    ncftp version 3.0 or greater (so that ncftpget may be used)
    The method used, either standard (local files) or ftp (remote files)
    depends on the following environment variables:

GRASP_FRAMEPATH             If using standard method, the directory in which
 (Always required)            input frames are located.
                              In the current implementation, ALL files contained
                              in this directory MUST be frame files.
                            If using ftp method, the local directory to which
                              remote tar archives are transferred for unpacking.
                              In the current implementation, this directory must
                              be initially empty.

GRASP_FTPCFG                This variable controls the method used by clean_signal
 (Required for ftp method)    for obtaining frames.
                            If unset, then the standard method is used
                            If set to "hsi", as in
                                 setenv GRASP_FTPCFG hsi
                              for instance, then the ftp method is used but,
                              instead of calling ncftpget to do the file transfer,
                              a local installation of hsi is used.
                            Otherwise, it should be set to the configuration file
                              needed by ncftpget for host, username, and password
                              information.  The format of this file is
                              HOST hostname
                              USER username
                              PASS password

                              For more information, see the ncftpget manpage.

GRASP_FTPFILE               If using ftp method, set to a file containing a list
 (Required for ftp method)    of tar archives to be obtained from the remote host
                              The format of the file is one filename per line.

GRASP_FTPDIR                If using ftp method, set to directory on remote host
 (Required for ftp method)    in which tar archives indicated in GRASP_FTPFILE are
```

```
                              located.

* Output Frames:

    There are two methods of printing output frames for clean_signal
    One is to print the output frames to a local file.  The other is
    to indicate a remote host on which output frames are to be stored.
    Use of the latter method requires a local installation of either ncftp
    version 3.0 or greater (so that ncftpput may be used) or hsi (in which
    case the remote server must be the HPSS storage system to which your
    version of hsi connects by default).  As in the input case, the method
    used depends on a few environment variables:

GRASP_CLEANPATH             If using standard method, the local directory
 (Always required)             to which output frames are to be stored.
                            If using ftp method, the remote directory to
                              which output frames are to be ftp'ed.

GRASP_FTPDESTCFG            If set, then ftp method is used for frame output
 (Required for ftp method)    If using ftp method, then it should be set to
                              either "hsi", in which case hsi is used to
                              perform the file transfer, or the configuration
                              file needed by ncftpput for host, username, and
                              password information.  See above (under
                              GRASP_FTPCFG) or the ncftpput manpage for more
                              information
```

2.1.3 Method

The method used by `clean_signal` to estimate and remove correlations from the signal channel is, as mentioned above, due to Allen, Hua, and Ottewill [2] Though the basic method is summarized briefly here, the paper in which it was originally proposed presents a much more thorough analysis.

To estimate the contamination of the signal due to environmental effects, three key assumptions are made. First, it is assumed that any such contamination is governed by a linear transfer function. What this assumption implies is that if y_i is the ith bin of the FFT of an environmental channel, then there exists a vector $\vec{r}$ of the same dimension as $\vec{y}$ such that the contamination in the ith bin of the signal channel FFT due to $\vec{y}$ is given by $r_i y_i$. Hence, if the transfer function $\vec{r}$ may be determined, then it may be used to remove the contamination due to the corresponding environmental channel from the signal. As a result of this first assumption, nonlinear contamination is not addressed.

Second, it is assumed that the linear transfer function, $\vec{r}$, is sufficiently smooth that the frequency range of the signal channel's Fourier transform may be divided into bands over which it is nearly constant. The length of the frequency bands used by `clean_signal` is an adjustable parameter which is indicated with the `-b` command-line switch. For a data sample of 256 seconds, the default band length, 128 bins, corresponds to 0.5 Hz. As a result of this assumption, sharp contaminations which result from "spikes" in the transfer function are not addressed. To be sure, only contaminations resulting from smooth (in frequency space) correlations are removed by the algorithm.

Finally, the third assumption is that the values of the transfer functions connecting the environmental channels to the signal channel in a given frequency band are such that, following the application of certain thresholding techniques, the power in the signal channel in that band is minimized when the environmental contaminations are removed. Without the reference to thresholding techniques, the implication of this assumption would be that any correlation, whether due to a real physical correlation or merely a random similarity between channels, is removed. Since it is undesirable to remove correlations of the latter type, a thresholding procedure designed to prevent the removal of so-called "false correlations" is applied prior to transfer function estimation. The problem of potentially removing "false correlations" is discussed in greater detail below.

With these assumptions, it is now possible to estimate the transfer functions connecting N environmental channels to the signal channel. Consider an arbitrary frequency band of length B, and represent the Fast Fourier Transform of each channel in that band by a B-dimensional vector. Let $\vec{x}^0$ denote the FFT of the signal channel and $\vec{x}^i$, $1 \leq i \leq n$, denote the FFTs of the environmental channels. By assumption, disregarding thresholding for the moment, the transfer function of the ith environmental channel, r^i, is a constant determined by the condition that the total power in the signal is minimized upon the removal of environmental contaminations. This is equivalent to choosing r^i to minimize the norm of $\vec{x}^0 - \sum_{i=1}^{N} r^i \vec{x}^i$. The variation in this norm with respect to r^i may be computed as the following, where $(,)$ is the standard Euclidean inner product:

$$-2\Re\left[\sum_{i=1}^{N} \delta r^i \left(\vec{x}^i, \vec{x}^0 - \sum_{j=1}^{N} r^j \vec{x}^j\right)\right]$$

Since this must vanish for arbitrary variations δr^i, the inner product in the above expression must vanish for all $j > 0$. This condition is rewritten below for emphasis:

$$\left(\vec{x}^i, \vec{x}^i - \sum_{j=1}^{N} r^j \vec{x}^j\right) = 0 \qquad i = 1, \ldots N$$

This is simply a statement that each environmental FFT, $\vec{v}^i$, must be orthogonal to the signal FFT following removal of environmental contaminations. An equivalent condition to that above is:

$$\left(\vec{x}^i, \vec{x}^0\right) = \sum_{j=1}^{N} r^j \left(\vec{x}^i, \vec{x}^j\right)$$

Defining the correlation matrix, C_{ij} by $C_{ij} = (\vec{x}^i, \vec{x}^j)$, this is equivalent to:

$$C_{i0} = \sum_{j=1}^{N} C_{ij} r^j$$

This is simply a matrix equation which may be solved for r^i using standard linear algebra techniques. `clean_signal` computes the correlation matrix from the FFTs and uses the CLAPACK routine `zhesv_()` to solve this equation, obtaining estimates for the environmental transfer functions. Before solving, though, it is necessary to make certain that the $N \times N$ matrix appearing on the right hand side is not singular, otherwise no solution will exist. Singularity is checked by determining those eigenvalues that are nearly zero with the CLAPACK routine `zheevx_()`. Whether or not an eigenvalue is "near zero" is determined by a threshold provided through the `-e` command-line switch. In order to remedy a singularity that may exist, the column whose projection onto the eigenvector corresponding to the "near-zero" eigenvalue is the greatest is removed, along with the row of the same index. This removes all of the correlations with one particular environmental channel, and hence has the effect of removing that channel from consideration. The eigenvalues are then checked again, and the process continues until the matrix on the right hand side is non-singular, at which point the solution is found. Once this is accomplished, the transfer function estimates are used to remove environmental contaminations and produce a “cleaned” signal channel.

In order for the above procedure to increase the signal to noise ratio within the signal channel, it is critical, as mentioned above, that nonexistent correlations are not removed. For instance, it was found by Allen, Hua, and Ottewill that if this scheme were applied blindly to a signal channel containing only Gaussian white noise, the norm of the signal channel FFT in each frequency band would be decreased by a factor of $1 - 1/B$ despite the fact that there are no correlations to remove. Allen, *et al* discuss several methods of preventing this, most of which involve setting the correlation matrix element between relatively uncorrelated channels to zero. The determination of whether two channels are relatively uncorrelated is made on the basis of their covariance, ρ_{ij}, which is defined to be:

$$\rho_{ij} = \sqrt{\frac{\|(\vec{x}^i, \vec{x}^j)\|^2}{(\vec{x}^i, \vec{x}^i)\,(\vec{x}^j, \vec{x}^j)}}$$

This is simply the cosine of the angle between $\vec{x}^i$ and $\vec{x}^j$. The initial procedure utilized by `clean_signal` involved establishing a minimum threshold for the covariance between two correlated channels and setting cross-correlations to zero when the corresponding covariances fell below this threshold. This presents a problem, though, since a determination of the appropriate covariance threshold is difficult in practice. If the covariance threshold is set too small, then the probabiliy of removing false correlations increases. More problematically, though, if the covariance threshold is set too large, then genuine cross-correlations in the correlation matrix are set to zero. When this occurs, r^i is no longer guaranteed to be such that a significant decrease in power occurs when used to remove the correlations. Worse yet, if the correlation matrix is sufficiently unstable, the arbitrary setting to zero of some of its elements may disrupt the determination of r^i to the point that noise is actually introduced into the signal channel. During the testing phase of `clean_signal`, this behavior was observed and it is for this reason that a more conservative thresholding procedure is presently used.

In order to avoid the removal of false correlations, `clean_signal` computes the covariance between the signal channel and each environmental channel in a given frequency band. If the covariance falls below a threshold, provided by the user with the `-c` switch, then the corresponding environmental channel is no longer considered in that band and the corresponding rows and columns are removed from the correlation matrix. The procedure then continues normally in that band as though the uncorrelated environmental channel were never present. The default value of the covariance threshold is $10/B$, where B is the length of each frequency band. This choice was motivated by the fact that Allen, *et al* determined the covariance between two statistically uncorrelated channels to be $1/B$.

Despite the thresholding procedure above, it is still likely that a strong, prominent signal will suffer at least a small reduction in power as a result of this procedure. The best means of estimating this reduction is through Monte Carlo techniques, and it is for this reason that `clean_signal` was outfitted with a signal injection feature.

2.2 Function: `gwsearch`

The `gwsearch` binary searches for evidence of periodic gravitational waves within data read from Frames and, if such evidence is found to exist, estimates the magnitude (in the source frame) of the plus and cross polarizations detected. Uncertainties are also estimated and, in addition, a number of files containing data for plotting are produced as well.

2.2.1 Command Line Usage

The user interacts with `gwsearch` through command line parameters and the setting of environment variables. The environment variables control Frame input, while the command line parameters control all other aspects of the function. The command line for gwsearch reads as follows:

```
Usage:  gwsearch  [options]

Options:

-c<string>     Sets name of channel to be obtained from frames to <string>
-f             Toggle FFT mode in which channel is assumed to contain an
                  FFT rather than raw data
-h             Help (prints this list)
-H             Help with environmental variables that must be set
-p<string>     Search for pulsar <string>.  Will look for polyco files of
                  the form <string>_polyco.dat produced by tempo.
                  Will look for right ascension and declination on the first
                  line of the file <string>.param.  These must be given in
                  order and MUST be expressed in RADIANS.
-v             Verbose
-w<string>     Tells gwsearch that the data being read from FRAME has been
                  windowed with the window function <string>.  It is CRUCIAL
                  that this is set properly
                  (DEFAULT = NONE)
                     Only other currently supported option is "HANNING"
-W<real>       Sets width of region in neighborhood of signal for which
                  data will be stored and printed to <real> Hz.


config_file is the name of the file containing names of frame files to be
            read and used by the gwsearch program
```

The `-c`, `-h`, `-v`, and `-w` are self-explanatory. The need for the `-w` option will become clear when the algorithm used by `gwsearch` is described in detail below. The other options will now be addressed.

FFT Mode

Use of the `-f` option toggles FFT mode, in which it is assumed that the channel indicated by the `-c` option contains an FFT rather than a raw signal. The FFT must be stored in the indicated channel as an array with the FFTW half-complex format. Denoting the array by A and its length by N, this format is to store the DC component in $A[0]$ (the assumption that we are dealing with an FFT of Real data is implicit), the real part of the jth component in $A[j]$, and the imaginary part of the j'th component in $A[N-j]$. In addition, the FFT coefficients are also assumed to be stored as doubles, a requirement that reflects the need to retain high precision when dealing with FFT's. This is not the case when not operating in FFT mode, as raw signal data may be stored in any of the standard data types. This is permitted since signal data is generally digitized, so storage in `int`'s and `short`'s is acceptable.

Pulsar Parameters

In order to effectively search for evidence of periodic gravitational waves, `gwsearch` needs sufficient information regarding the target pulsar to determine the expected frequency and amplitude modulation of the corresponding interferometer signal. This information is obtained from auxiliary files whose names are based on the pulsar name provided by the `-p` switch. To determine frequency modulation, `gwsearch` depends on the same output files from the TEMPO timing code needed by `clean_signal` for its signal injection feature (see section 2.1.1 above). The TEMPO output file corresponding to the pulsar indicated with the `-p` command-line switch must be named <pulsar name>_polyco.dat.

To determine the amplitude modulation, `gwsearch` needs the righta ascension and declination of the target pulsar. These should be contained within the user-created file <pulsar name>.param. The format of this file is identical to that needed by `clean_signal` (see section 2.1.1 above), with the right ascension and declination being given, in radians and in that order, on the file's first line.

Neighboring Frequency Bins

In order to determine whether or not a signal detection has taken place, the frequency bin in which the sought-after signal is expected to be found, hereafter referred to as the "signal bin", must be compared to adjacent bins, which may be used to estimate channel noise. For this reason, `gwsearch` applies its algorithm not only to the signal bin, but also to neighboring bins within a distance, in Hz, from the signal bin indicated by the `-W` command-line switch. As a result, this interval is **ALWAYS** symmetric. If the interval extends beyond the range of validity of the FFT (either greater than the Nyquist frequency or less than 0), then an error message is printed warning that the effectiveness of the algorithm may be affected. The manner in which the neighboring bins are used by `gwsearch` is described in more detail below.

2.2.2 Environment Variables

The input to `gwsearch` is controlled by the same environment variables which control the input to `clean_signal`, as described in section 2.1.2. Note that only the portion of this section pertaining to input applies to `gwsearch`, since `gwsearch` does not produce any frame files for output.

2.2.3 Method

Signals within interferometer data that correspond to periodic gravitational wave sources exhibit both frequency modulation, due to the relative motion of the earth and source, and amplitude modulation, due to the varying sensitivity of the interferometer to the two different gravitational wave polarizations as the earth rotates. The presence of these effects complicates data analysis, which must take the into account. Failure to properly treat frequency modulation may lead to decreased sensitivity due to the possibility that the signal frequency may migrate from one bin to another, while failure to properly account for amplitude modulation could lead to a similar decrease over long time intervals due to the potential flip in the detector sensitivity's sign during its oscillation period. In order to prevent these effects from decreasing the sensitivity of the search, possibly to levels at which potential signal detections are completely washed out, `gwsearch` uses a method which is based upon the source's frequency and amplitude modulation. The side effect to this is that `gwsearch` looks for gravitational radiation from a fixed, target pulsar and, hence, it is not well adapted to an all-sky search.

In order to address frequency and amplitude modulation, `gwsearch` must first estimate the time evolution of these quantities for the pulsar in question. The method by which these estimations are accomplished will thus be addressed first, followed by a detailed description of the search algorithm.

Frequency Modulation

As mentioned several times above, the determination of a given pulsar's frequency at the time of observation is done by the TEMPO timing code. The GWCLASS binaries merely read these output files and use the information contained therein. What TEMPO produces when the `-z` switch is used is a file, usually called "polyco.dat", which contains a series of polynomial approximations to the pulsar frequency modulation as a function of time. For each polynomial approximation, there is a corresponding series of lines in this file. The first two lines of each such series contain a number of parameters, including the length of the time interval during which the polynomial approximation is valid, and the median time of that interval. The following lines contain the polynomial coefficients, with three listed per line, in scientific notation. GWCLASS binaries read these polynomial coefficients and use them to determine the pulsar frequency at any given point in time. In order to use these polynomial coefficients, though, GWCLASS binaries must first convert the GPS times at which frequencies are to be computed to Modified Julian Date, or MJD, as the Tempo polynomial approximation requires the time to be expressed in the latter units.

Before continuing, it is **CRUCIAL** to note that GWCLASS expects the format of the polynomial coefficients to be nDp where n is a floating point number and p is an integer exponent. Contrast this to the usual scientific notation format, nEp. GWCLASS was written to look for a D rather than an E because the Tempo binary first used by the author, which had been compiled with a relatively old fortran compiler, wrote scientific notation in this manner. In addition, use of the D's was found to be advantageous because it permits the argument and exponent to be read separately (the C default is to automatically read $1E10$ as 10, etc). For convenience, the GWCLASS distribution includes a `perl` utility script entitled `e2d` which reads the Tempo output files and converts E's to D's when necessary.

Amplitude Modulation

In addition to the characteristic frequency modulation which impacts a number of astrophysical observations, gravitational wave detection through interferometric techniques must also cope with amplitude modulation in the form of varying sensitivities to the two different polarization states. The reason for this is that the interferometer acts as a polarization filter, detecting only that component of the incident radiation aligned with the interferometer arms, whose relative orientation to the source is constantly changing. Thus, the interferometer response to a steady signal emitted from an astrophysical source will be a function of time which must be determined for effective detection. A general discussion of this problem may be found in a Maple worksheet by Anderson, *et al*[3] A simplified treatment, which differs somewhat from that of Anderson, *et al*, is presented here.

The response of the interferometer is most easily expressed in a reference frame, termed the "detector frame," with x and y axes pointing along the arms and z axis pointing toward the zenith. For a gravitational wave tensor, $h_{\mu\nu}^{(D)}$, expressed in this frame, the response takes the simple form $\frac{1}{2}(h_{11}^{(D)} - h_{22}^{(D)})$. On the other hand, since the detector frame is in motion relative to the source, the incident gravitational wave is most easily expressed in a "source frame," having origin at the source, z axis pointing toward the earth's center, and y axis pointing due north along a line of constant right ascension. In addition, it is convenient to define the "plus" and "cross" polarizations in this frame by choosing to identify them with the usual TT gauge basis tensors there:

$$\hat{e}_{+}^{(S)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \hat{e}_{\times}^{(S)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The $\left(\hat{e}_{+}^{(S)}\right)_{0j}$ and $\left(\hat{e}_{\times}^{(S)}\right)_{j0}$ components, which are all vanishing, have been omitted since only spatial rotations shall be considered. Given an incident gravitational wave of the form $h^{(S)} = h_{+}^{(S)}\hat{e}_{+}^{(S)} + h_{\times}^{(S)}\hat{e}_{\times}^{(S)}$ in the source frame, it is necessary to express it in the detector frame to determine the detector response. This may be accomplished by a translation, whose impact on h is trivial, followed by a series of rotations. Let R denote the passive rotation which transforms the coordinate representation of a fixed vector in the translated source frame to its coordinate representation in the detector frame and note that its rows express the detector frame basis unit vectors in terms of the source frame basis unit vectors. Once R is determined, the matrix representation of h in the detector frame, $h^{(D)}$, may be expressed in terms of its matrix represention in the source frame, $h^{(S)}$, as $Rh^{(S)}R^T$, or $h_{rs}^{(S)}R_{ir}R_{js}$. The determination of R may be fairly complicated, though, depending upon the sophistication of the earth model used. In addition, since the two frames are in motion relative to one another, R will be time-dependent. For these reasons, it is often easier to consider an intermediate frame, the "Earth fixed frame," with origin at the earth's center, x axis in the equatorial plane pointing toward the Prime Meridian, and y axis in the equatorial plane pointing toward 90° East longitude. The reason for this is that the transformation from source frame to Earth fixed frame may be easily expressed in terms of Euler angles, while the possibly complicated transformation from Earth fixed frame to the detector frame is time-independent. What's more, the latter need only be determined once for each detector and, once determined, may be used in the computation of sensitivities for any periodic source in the sky!

Now, let A denote the matrix of a passive transformation from the source to Earth fixed frame, and let B denote the matrix of a passive transformation from the Earth fixed to the detector frame. In terms of A and B, $h^{(S)}$ may now be expressed in the detector frame as $h^{(D)} = (BA)h^{(S)}(BA)^T$, or $h_{ij}^{(D)} = B_{il}A_{lr}h_{rs}^{(S)}A_{ks}B_{jk}$. In terms of A and B, the sensitivity is given by the following:

$$\begin{aligned} \frac{1}{2}\left(h_{11}^{(D)} - h_{22}^{(D)}\right) &= \frac{1}{2}\left(B_{1l}A_{lr}h_{rs}^{(S)}A_{ks}B_{1k} - B_{2l}A_{lr}h_{rs}^{(S)}A_{ks}B_{2k}\right) \\ &= \frac{1}{2}\left(B_{1l}B_{1k} - B_{2l}B_{2k}\right)A_{lr}h_{rs}^{(S)}A_{ks} \end{aligned}$$

Now, define the response tensor, d^{kl}, as:

$$d^{kl} = \frac{1}{2}\left(B_{1l}B_{1k} - B_{2l}B_{2k}\right)$$

Since $A_{lr}h_{rs}^{(S)}A_{ks}$ is simply the matrix representation of h in the Earth fixed frame, denoted $h^{(E)}$, the sensitivity in the detector may be expressed as the contraction $h_{ij}^{(E)}d^{ij}$. All of the necessary information concerning a given detector's location, including the earth model used for its determination, are contained within the response tensor, which need only be determined once per detector. The response tensors for many of the active gravitational wave interferometers obtained with the WGS-84 Elliptical Earth model, as well as methods for computing them, are listed in the previously cited Maple worksheet. At present, `gwsearch` uses only the response tensor for the 40 meter prototype interferometer at the California Institute of Technology. To use `gwsearch` to analyze data taken from other instruments will require additional response tensors, which are not present in the code at this time.

Once the response tensor is known, the sensitivity in the detector may be computed after $h^{(E)}$ is determined. This will eventually be accomplished through the use of Euler angle rotations, but first the expression for $h^{(E)}$ is further analyzed. Let x_s, y_s, z_s denote the axes in the source frame and let x_e, y_e, z_e denote the axes in the Earth fixed frame. Recalling the above definition of A, the unit polarization tensors $\hat{e}_+^{(S)}$ and $\hat{e}_\times^{(S)}$ take the following forms in the Earth fixed frame:

$$\begin{aligned}\left(\hat{e}_+^{(E)}\right)_{ij} &= \left(A\hat{e}_+^{(S)}A^T\right)_{ij} \\ &= A_{ir}\left(\hat{e}_+^{(S)}\right)_{rs}A_{js} \\ &= A_{i1}A_{j1} - A_{i2}A_{j2}\end{aligned}$$

$$\begin{aligned}\left(\hat{e}_\times^{(E)}\right)_{ij} &= \left(A\hat{e}_\times^{(S)}A^T\right)_{ij} \\ &= A_{ir}\left(\hat{e}_\times^{(S)}\right)_{rs}A_{js} \\ &= A_{i1}A_{j2} + A_{i2}A_{j1}\end{aligned}$$

This expression may be understood further by noting that the 1st column of A, A_{i1}, consists of those coefficients which express the source frame unit vectors in the Earth fixed frame basis. Thus, letting $\{\hat{x}_s, \hat{y}_s, \hat{z}_s\}$ denote the source frame basis unit vectors and $\{\hat{x}_e, \hat{y}_e, \hat{z}_e\}$ denote the Earth fixed frame unit basis unit vectors, the plus polarized unit tensor in the Earth fixed frame becomes the outer product of $\hat{x}_s$ with itself less the outer product of $\hat{y}_s$ with itself, expressed in the $\{\hat{x}_e, \hat{y}_e, \hat{z}_e\}$ basis. Similarly, the cross polarized unit tensor in the Earth fixed frame becomes the sum of the outer product of $\hat{x}_s$ with $\hat{y}_s$ and the outer product of $\hat{y}_s$ with $\hat{x}_s$, again expressed in the Earth fixed frame basis. Since a general h may be written as a linear combination of these basis tensors, the sensitivity may be determined once expressions for $\hat{x}_s$ and $\hat{y}_s$ in the Earth fixed frame basis are found. This is accomplished by constructing the passive rotation A defined above. To do this, the Euler angles Θ and Φ, illustrated in figure 2.1, are used. Generally, there is a third Euler angle Ψ involved in this transformation as well, but the source frame has been chosen in such a manner that this angle is 0. Anderson *et al* give a more general treatment which includes the possibility that Ψ is nonzero. The angle Θ is defined as the smallest angle between z_s and z_e, while the angle Φ is the smallest angle between x_s and x_e, both of which lie in the $x_e y_e$ plane. The required composite rotation is given by the following product of Euler rotation matrices:

$$\begin{aligned}A &= \begin{pmatrix}1 & 0 & 0 \\ 0 & \cos\Theta & \sin\Theta \\ 0 & -\sin\Theta & \cos\Theta\end{pmatrix}\begin{pmatrix}\cos\Phi & \sin\Phi & 0 \\ -\sin\Phi & \cos\Phi & 0 \\ 0 & 0 & 1\end{pmatrix} \\ &= \begin{pmatrix}\cos\Phi & \sin\Phi & 0 \\ -\sin\Phi\cos\Theta & \cos\Phi\cos\Theta & \sin\Theta \\ \sin\Phi\sin\Theta & -\cos\Phi\sin\Theta & \cos\Theta\end{pmatrix}\end{aligned}$$

The components of the $\hat{x}_s$ and $\hat{y}_s$ unit vectors expressed in the Earth fixed frame basis are now easily read off as the first and second rows of the above matrix. `gwsearch` uses this result[1], computes the appropriate outer products to determine $h^{(E)}$, and contracts with the response

[1]**IMPORTANT NOTE:** the code for amplitude determination by `gwsearch`, contained in the file `amplitude.c`, uses the variables `phi` and `theta` to refer to the azimuth, ϕ, and altitude, θ, respectively, rather than the Euler angles Φ and Θ. The relation between these two sets of angles is given by $\Theta = \pi - \theta$ and $\Phi = \phi - \pi/2$. See the Maple worksheet of Anderson, *et al* for more information.

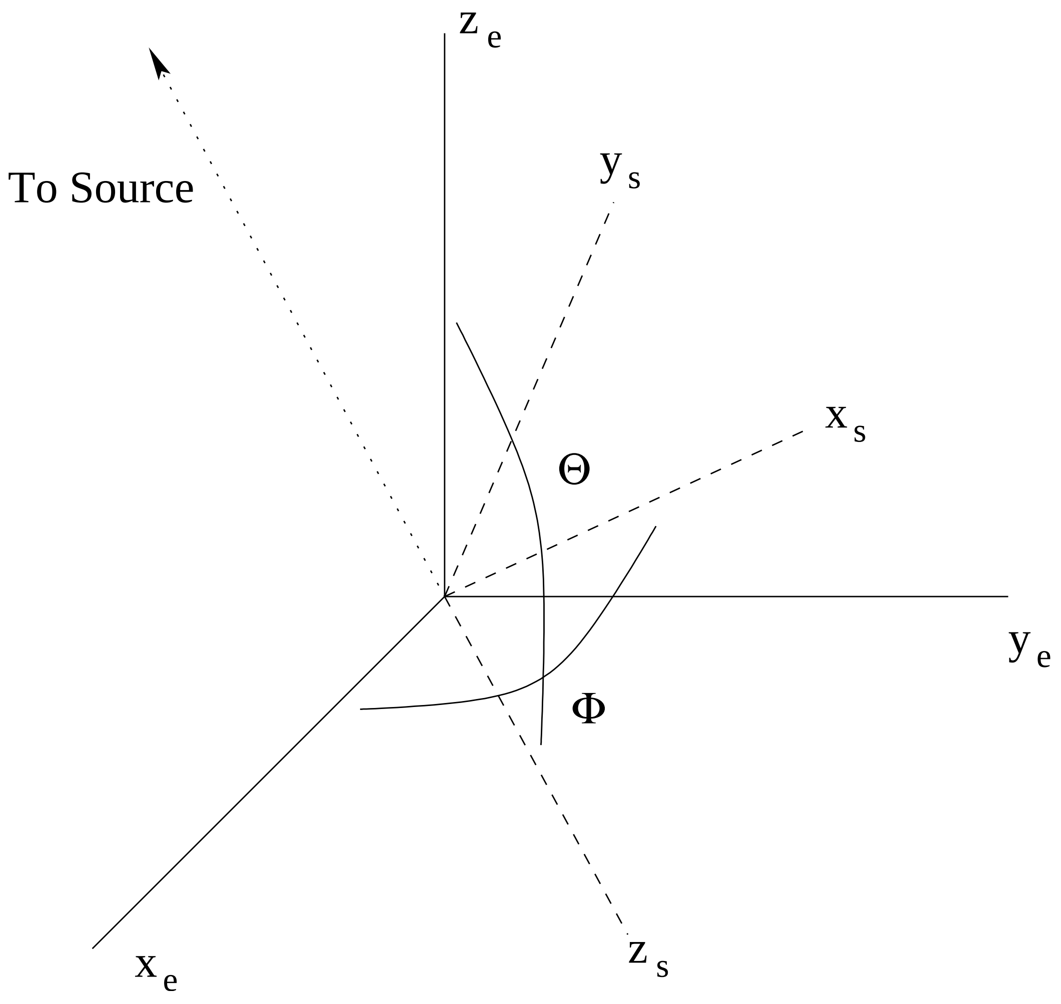

Figure 2.1: Euler rotation angles

tensor to obtain the sensitivity to the plus and cross polarizations. The only remaining task is the determination of the relation between the Euler angles, Φ and Θ, and the source's right ascension, α, its declination, δ, and the time of observation, t, expressed in Greenwich Mean Sidereal Time. In what follows, it is assumed that α, δ, t, Θ, and Φ are all expressed in radians. It is clear from figure 2.1 that $\Theta + \left(\frac{\pi}{2} - \delta\right) = \pi$ so $\Theta = \delta + \frac{\pi}{2}$. The relation between α, t, and Φ is less clear. To obtain it, first note that since x_s lies in the $x_e y_e$ plane, the $y_s z_s$ plane must be perpendicular to the $x_e y_e$ plane. This implies that the projection of $-z_s$, which points toward the source, onto the $x_e y_e$ plane is along the line of intersection between the $x_e y_e$ plane and $y_s z_s$ plane and hence, lying in the $y_s z_s$ plane, is perpendicular to x_s. Now, $\alpha - t$ is the angle between this projection and the x_e axis, so $\alpha - t = \Phi + \frac{\pi}{2}$, or $\Phi = \alpha - t - \frac{\pi}{2}$. Plugging these relations into previous results yields the method by which `gwsearch` computes amplitude sensitivities.

Search Algorithm

As previously mentioned, an effective search for periodic gravitational wave signals must account for the frequency and amplitude modulation caused by the relative motion and subsequent changes in relative orientation of the source and detector. The method presented here is based upon the assumptions that both forms of modulation are sufficiently slow that there exist time intervals over which the amplitude and frequency of a gravitational wave signal may be taken as constant. This permits a time-frequency analysis, in which the time evolution of the power within a particular frequency bin may be studied by constructing a time series for the time evolution of the power within that bin. At this point, one method of approach involves constructing a similar time series, with one coefficient taken from each DFT but, rather than using the same frequency bin each time, the index of the bin that is taken from a given DFT is shifted to account for the shift in frequency of the source. At this point, the time series follows the power within the bin in which the source frequency may be found and hence, to some extent, the evolution of the signal power itself, provided that it is not completely obscured by noise. In fact, a number of such time series, each utilizing the same fixed shift in bin index for each DFT, may be constructed, from which a mean may be computed. The mean in the series corresponding to the signal frequency bin may then be compared to the rms of the means within the series from neighboring bins to determine whether or not a signal is present. This approach would be effective when searching for a relatively strong frequency modulated signal, but fails to account for amplitude modulation and, in addition, makes use only of the magnitude of the DFT coefficients, neglecting their phase. When searching for weak signals, using the phase information encoded within the DFT may lead to an increase in the search's sensitivity. It is for this reason that a slightly more complicated procedure, which also incorporates amplitude modulation, is used by `gwsearch`.

In order to incorporate phase information into a gravitational wave search, one means of modifying the technique described above is the following. Rather than determining the average magnitude of the Fourier coefficients in a "time series" of appropriately shifted frequency bins, what is computed instead is the sum of the Fourier coefficients themselves which, owing to the fact that they are complex and may hence be viewed as vectors in $\mathbb{R}^2$, gives rise to cancellation if they are not properly aligned. In order to prevent Fourier coefficients due to a physical signal from suffering destructive interference during such a procedure, though, they must first be phase shifted to account for the difference in phase between the coefficients within the signal bins of successive time intervals. This problematic phase difference, which must be accounted for, is caused by the fact that Fourier transforms from two different time intervals are computed with respect to two different reference, or "starting" times. For instance, a signal of the form $A\cos(\omega(t)t + \phi)$ has phase ϕ relative to $t = 0$ while having phase $\omega(t_0)t_0 + \phi$ relative to some other time $t = t_0$. Unfortunately, due to accumulated phase caused by frequency modulation, the phase shift between DFT's from successive time intervals does not take the simple form $\omega(t_0)t_0$ despite the fact that this expression correctly identifies the phase difference between the actual signals.

The manner in which `gwsearch` avoids these problems is as follows. Within each time interval, the DFT of an artificially constructed pure signal of unit amplitude which exhibits the expected frequency modulation is computed. The phase shift, Ψ, of the DFT coefficient within the signal bin relative to that of the first time interval is then determined and the DFT of the data channel within a neighborhood of the signal bin is multiplied by $e^{-i\Psi}$. This effectively cancels the shift in phase between the signal DFT in the current interval and that in the first interval. The result is that the DFT coefficients corresponding to an astrophysical signal will be aligned with one another throughout all time intervals under consideration with phase equivalent to that relative to the beginning of the first interval. So, a signal whose DFT is purely real in the first interval will have its DFT coefficients rotated in subsequent time intervals such that they are purely real as well, and so on. On the other hand, nonastrophysical noise will not in general be aligned in the same manner, as such noise generally fails to follow the same frequency modulation as an astrophysical source, and hence fails to exhibit the same DFT coefficient phase shifts. This will lead to destructive interference when they are added, giving rise to an increase in signal-to-noise ratio.

In addition to phase shift problems, though, we must also deal with leakage from the signal bin caused by the discrete nature of the DFT. In order to effectively estimate the strength of a detected gravitational wave signal, the data channel DFT is normalized through division by the magnitude of the signal bin coefficient of the articially constructed unit amplitude signal described above. This action has no effect on the signal-to-noise ratio since it is merely a global rescaling of the DFT.

If the interferometer response to an incident gravitational wave signal displayed no amplitude modulation, then the above procedure could be applied without difficulty. The phase shifts would ensure that the signal bin DFT coefficients corresponding to the target pulsar add constructively while destructive interference decreases the noise in neighboring bins. With the rescaling, the average of the signal bin would yield a good estimate of the gravitational wave strength, and the rms in the neighboring bins a good estimate of the associated uncertainty. Unfortunately, the situation is not this simple, so the above method must be modified even further. The first steps, though, namely the phase shifting and rescaling, remain unchanged. It is merely necessary to replace the process of adding DFT coefficients from successive intervals with something more sophisticated that takes amplitude modulation into account.

Consider the vector $\vec{x}$, whose ith component is the DFT coefficient of the signal bin within the ith time interval. Now, let $\vec{F}^{+}$ and $\vec{F}^{\times}$ denote the N-dimensional vectors whose ith components contain the true sensitivities of the interferometer to a unit plus polarized gravitational wave and a unit cross polarized gravitational wave, respectively, at the median time of the ith time interval. Since the sensitivity is assumed to be relatively constant within any given time interval, a gravitational wave of the form $h_{+}\hat{e}^{(+)} + h_{\times}e^{i\phi}\hat{e}^{(\times)}$, where h_{+} and $h_{\times}$ are real and ϕ denotes the phase difference between the plus and cross polarization components, gives rise to the DFT coefficient $x_i = h_{+}F_i^{+} + h_{\times}F_i^{\times}e^{i\phi}$, within the ith time interval, following application of the above procedure[2]. The reason for this is that the rescaling maintains the amplitudes h_{+} and $h_{\times}$, while the phase shifting assures that there is no phase difference between DFT coefficients from different time intervals, x_i and x_j. Equivalently, this rescaling and phase shifting assures that the the plus and cross polarized components of x_i have the same amplitude and phase as the respective components of x_j. This only holds in the ideal case, though, in the absence of noise. To treat the actual case, let $\vec{g}$ denote an N-dimensional complex vector whose ith component, written as $g_j \exp(i\psi_j)$ with g_j and ψ_j both real, represents the FFT of the data channel less that of a pure signal. It is assumed that the noise represented by $\vec{g}$, and hence g_j and ψ_j, is random in nature. Including the effects of this noise, the vector $\vec{x}$ properly takes the following form:

$$x_j = h_{+}F_j^{+} + h_{\times}F_j^{\times}e^{i\phi} + g_j e^{i\psi_j}$$

To estimate h_{+} given $\vec{x}$, $\vec{F}^{+}$, and $\vec{F}^{\times}$, first look at the weighted sum, $x_j\vec{F}_j^{+}$, where values of x_j originating in time intervals of relatively high sensitivity to plus polarized gravitational waves are given greater emphasis. The result of this weighted sum is the following:

$$x_j F_j^{+} = h_{+}\left(F_j^{+}F_j^{+}\right) + h_{\times}\left(F_j^{\times}F_j^{+}\right)e^{i\phi} + g_j e^{i\psi_j}F_j^{+}$$

The norm of $\vec{F}^{(+)}$ is much greater than its projection onto $\vec{F}^{(\times)}$ so, provided $h_{+}F^{+}$ is larger than g_j, which is required for a signal detection, x_j is dominated by the $h_{+}\left(F_j^{+}F_j^{+}\right)$ term. For this reason, the estimated value of h_{+}, termed $h_{+}^{?}$, is taken to be the magnitude of the complex number $x_j\vec{F}^{+}/|\vec{F}^{+}|^2$. In terms of the actual values, this implies that the estimate, $\left(h_{+}^{?}\right)^2$, is given by:

[2]Actually, this expression is correct only up to an arbitrary phase, determined by the starting time of the first interval, which is fixed for all subsequent intervals. Since this constant phase simply multiplies the entire response vector, it may be neglected.

$$
\begin{aligned}
\left(h_+^?\right)^2 &= h_+^2 + h_\times^2\frac{\left(F_j^\times F_j^+\right)^2}{\left(F_k^+F_k^+\right)^2} + \frac{\left(g_jF_j^+\right)^2}{\left(F_k^+F_k^+\right)^2} + 2\frac{\left(F_j^\times F_j^+\right)}{\left(F_k^+F_k^+\right)}h_+h_\times\cos\phi \\
&+2\frac{F_j^+g_j\cos\psi_j}{\left(F_k^+F_k^+\right)}h_+ + 2\frac{\left(F_j^\times F_j^+\right)}{\left(F_k^+F_k^+\right)^2}F_r^+g_r\cos\left(\phi-\psi_r\right)h_\times
\end{aligned}
$$

Since ψ_j is a random phase and F_j^+ is a smoothly varying function of time, one might expect the fluctuations of $\cos\psi_j$ and $\cos\left(\phi-\psi_j\right)$ to cause $F_j^+g_j\cos\psi_j$ and $F_r^+g_r\cos\left(\phi-\psi_r\right)$ to become negligibly small if sufficiently many intervals are considered. One must be careful, though, because these terms are only first order in the noise factors, g_r, while the only other noise-dependent term is of second order in the g_r. For this reason, all terms are kept to yield a conservative estimate of the error.

The above expression may be rewritten as follows, where $(,)$ is the standard Euclidean inner product:

$$
\begin{aligned}
\left(h_+^?\right)^2 &= h_+^2 + \frac{\left(\vec{g},\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + h_\times^2\frac{\left(\vec{F}^\times,\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+h_\times\cos\phi \\
&+2\frac{\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+\cos\psi_j + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)^2}h_\times\cos\left(\phi-\psi_j\right)
\end{aligned}
$$

Thus, in the limit in which many time intervals are considered, the estimated value of $\left(h_+^?\right)^2$ differs from the actual h_+^2 by an amount $\delta\left[\left(h_+^?\right)^2\right]$, given by the following:

$$
\begin{aligned}
\delta\left[\left(h_+^?\right)^2\right] &= \frac{\left(\vec{g},\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + h_\times^2\frac{\left(\vec{F}^\times,\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+h_\times\cos\phi \\
&+2\frac{\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+\cos\psi_j + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)^2}h_\times\cos\left(\phi-\psi_j\right)
\end{aligned}
$$

The terms involving the overlap of $\vec{F}^+$ and $\vec{F}^-$ would be zero if the sensitivity functions due to the two independent polarizations were orthogonal, but this is not the case in general. This is a manifestation of the fact that while the two independent polarization states are orthogonal to one another, the components which are detected by the interferometer, which acts as a polarization filter, generally are not.

It is possible to determine $\vec{F}^+$ and $\vec{F}^\times$ with the algorithm of Anderson, *et al* [3], while an estimation of the noise is obtained by performing the same procedure as that applied to the signal bin, namely phase shifting, rescaling, projecting onto $\vec{F}^+$, and dividing by the norm squared of $\vec{F}^+$, on the neighboring frequency bins. Very near the signal bin, the neighboring bins will contain signal leakage, but if a sufficiently large neighborhood is used, then the rms value of $\left(\vec{x},\vec{F}^+\right)/\left(\vec{F}^+,\vec{F}^+\right)^2$ within that neighborhood will provide a good estimate for the noise term of $\delta\left[\left(h_+^?\right)^2\right]$. Unfortunately, the true values of h_+ and $h_\times$, as well as the phase difference, ϕ, between the plus and cross polarized components, are generally unknown. Thus, the following procedure is used for determining the error in $\left(h_+^?\right)^2$ by gwsearch. Presumably the h_+^2 estimate will be largely due to the existence of a plus polarized gravitational wave, but in addition it will be artifically increased by the presence of a cross polarized wave. Similarly, the $h_\times^2$ estimate will be inflated if a plus polarized component exists. Because of this, the estimated values of h_+^2 and $h_\times^2$ are used in the expression for $\delta\left[\left(h_+^?\right)^2\right]$ since, being larger than the actual values, they will produce an overestimate, rather than an underestimate, of the error. In addition, the trigonometric factors are all set to one to obtain a conservative estimate. Once the expression for $\delta\left[\left(h_+^?\right)^2\right]$ is obtained, standard error propagation techniques are used to estimate $\delta h_+^?$. The preceding algorithm may also be applied for $h_\times$. The resulting expressions for both h_+ and $h_\times$ are summarized below for clarity:

$$\left(h_+^?\right)^2 = \frac{\left(\vec{x},\vec{F}^+\right)}{\left(\vec{F}^+,\vec{F}^+\right)^2} \qquad \left(h_\times^?\right)^2 = \frac{\left(\vec{x},\vec{F}^\times\right)}{\left(\vec{F}^\times,\vec{F}^\times+\right)^2}$$

$$\begin{aligned}
\delta\left[\left(h_+^?\right)^2\right] &= \frac{\left(\vec{g},\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + h_\times^2\frac{\left(\vec{F}^\times,\vec{F}^+\right)^2}{\left(\vec{F}^+,\vec{F}^+\right)^2} + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+h_\times\cos\phi \\
&\quad +2\frac{\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)}h_+\cos\psi_j + 2\frac{\left(\vec{F}^\times,\vec{F}^+\right)\left(\vec{F}^+,\vec{g}\right)}{\left(\vec{F}^+,\vec{F}^+\right)^2}h_\times\cos\left(\phi-\psi_j\right)
\end{aligned}$$

$$\begin{aligned}
\delta\left[\left(h_\times^?\right)^2\right] &= \frac{\left(\vec{g},\vec{F}^\times\right)^2}{\left(\vec{F}^\times,\vec{F}^\times\right)^2} + h_+^2\frac{\left(\vec{F}^+,\vec{F}^\times\right)^2}{\left(\vec{F}^\times,\vec{F}^\times\right)^2} + 2\frac{\left(\vec{F}^+,\vec{F}^\times\right)}{\left(\vec{F}^\times,\vec{F}^\times\right)}h_\times h_+\cos\phi \\
&\quad +2\frac{\left(\vec{F}^\times,\vec{g}\right)}{\left(\vec{F}^\times,\vec{F}^\times\right)}h_\times\cos\psi_j + 2\frac{\left(\vec{F}^+,\vec{F}^\times\right)\left(\vec{F}^\times,\vec{g}\right)}{\left(\vec{F}^\times,\vec{F}^\times\right)^2}h_+\cos\left(\phi-\psi_j\right)
\end{aligned}$$

When run, gwsearch determines whether the estimates of h_+ and $h_\times$ are greater than 5 times the noise terms (those that depend on the g_r), and flags a potential detection if this is the case. Then, the estimated values of h_+ and $h_\times$, along with their associated uncertainties, are printed to screen. In addition, gwsearch produces two files, <pulsar name>.in and <pulsar name>.quad, which contain data for plotting. The former contains triples of the form "t f A", where t is the median time of a given interval in MJD, f is the deviation from the signal frequency, and A is the in-phase component of the appropriately phase-shifted and rescaled DFT coefficient, which is defined as the DFT coefficient's projection onto the line in $\mathbb{R}^2$ defined by the reference DFT coefficient. The latter contains similar triples, with A denoting the quadrature-phase component defined as the projection of the DFT coefficient onto the orthogonal compliment of the line defined by the reference DFT coefficient. These files may be plotted easily with Matlab or Maple in order to visualize the DFT coefficients of the data channel after being properly phase shifted and rescaled. If a strong signal is present, then a steady peak at $f = 0$, which exhibits gradual amplitude modulation, should be evident.

2.3 Function: `dump_frame`

The `dump_frame` utility reads data from a user-indicated channel from a user-provided frame and dumps the results into an ASCII text file. This utility is useful for obtaining channels which store data FFT's, at which point the printed ASCII text file contains a power spectrum of the indicated FFT channel. If FFT mode is not toggled, the output ASCII files bear names of the form `<channel name>_<GPS start time>.txt`. The output file contains a list of ordered pairs of the form (t, x), where t is the number of seconds into the Frame, and x is the value of the indicated channel at that time. On the other hand, if FFT mode is toggled, then the output file is named `<channel name>_pow_<GPS start time>.txt` and contains ordered pairs (f, y) with f being a frequency, in Hertz, and y the power within the channel at that frequency.

2.3.1 Command Line Usage

The usage for `dump_frame` reads as follows:

```
Usage: dump_frame [options] [frame file]

[frame file] should be the name of a frame whose data is to be
dumped

Options:
   -c<string>    Dumps data for channel <string> to ascii files
                    (DEFAULT = 'IFO_DMRO')
   -f            Turns on 'fft' switch, in which indicated channel is
                    assumed to contain the fft of a real array stored
                    in the fftw half-complex format (see fftw user's guide
                    (DEFAULT is OFF)
```

As a final note, the half-complex format of FFT's is discussed not only in the FFTW user's guide, but also in section 2.2.1 under the 'FFT Mode" heading above.

2.4 Function: gps2gmst

The function `gps2gmst` converts a user-provided time in GPS seconds to Greenwich Mean Standard Time, with the result expressed in radians.

2.4.1 Command Line Usage

The `gps2gmst` command line usage reads as follows:

```
Usage:  gps2gmst <gps seconds>
```

2.4.2 Method

The algorithm used by `gps2gmst`, begins with a conversion from GPS seconds to MJD (Modified Julian Date) whose method is described in section 2.5. Then, it applies an expression from page S15 of the 1984 Astronomical Almanac, yielding the difference between GMST and the UT1 time, which is equivalent to the angle, in arcseconds, between the Greenwich zenith and the line of 0 right ascension, and midnight (0 hours). Once this is determined, and converted to radians, the product of 2π and the fraction of a day which has passed at the indicated time need only be added to obtain the GMST.

The Astronomical Almanac expression is a third degree polynomial, $A + BT + CT^2 + DT^3$, with $T = (M - 51544.5)/36525$ and M defined as the date in MJD. The coefficients have the following values:

$$\begin{aligned} A &= 24110.54841 \\ B &= 8640184.812866 \\ C &= 0.093104 \\ D &= -0.0000062 \end{aligned}$$

2.5 Function: gps2mjd

The function gps2mjd converts a time given in GPS seconds to its equivalent MJD (Modified Julian Date).

2.5.1 Command Line Usage

The command line usage for gps2mjd reads:

```
Usage: gps2mjd <gps_seconds>
       Returns equivalent MJD (Modified Julian Date)
```

2.5.2 Method

The input GPS time is first converted to UTC calendar time with the aid of the GRASP function gtime(). At this point, the number of days since Midnight, December 31, 1995 is determined and added to the MJD of that date, 50083. The appropriate fraction of a day is then computed and added.

2.6 Function: lockcat

The function lockcat reads data from an input stream of Frame files and compiles a catalog of locked sections. The results are printed to an output file, lock.txt by default, with each line containing the starting GPS time, ending GPS time, and length, in seconds, of a locked section. This function is useful for streamlining the list of Frame files sent to clean_signal to avoid wasted computing time. A number of the utility scripts, including process_listing and make_tar_list aid in this task as well. These will be described further in chapter 3.

2.6.1 Command Line Usage

The command line usage for lockcat reads:

```
Usage :  lockcat [options]
Options:
-c<string>       Sets name of channel to read while searching for locked
                    sections to <string>
                    (DEFAULT = IFO_DMRO)
-f<string>       Sets name of file to which lock info is printed to <string>
                    (DEFAULT = lock.txt)
-h               Prints this screen
-n<integer>      Sets number of points to be read at a time to <integer>
                    Note: Locked sections of length smaller than <integer>
                    may be missed.
                    (DEFAULT = 16384)
-v               Verbose
```

Input is handled through the same environment variables utilized by `clean_signal` and `gwsearch`. Their usage may be found in section 2.1.2.

2.6.2 Method

This routine depends upon the GRASP function `fget_ch()` to read Frame data, and makes use of the returned values `fgetoutput.lostlock_gps` and `fgetoutput.lastlock_gps` which, after reading some data, contain the GPS times corresponding to the most recent lock loss and the most recent reacquisition of lock, respectively. These values are merely printed to the output file at the proper times.

It is critical to note the method by which the modified version of `fget_ch()` determines whether or not the interferometer is in lock. This is accomplished by looking at the lock channel, `IFO_Lock`. When the value in this channel lies between two constants, `locklow` and `lockhi`, the interferometer is said to be in lock. Some Frames have a `locklow/lockhi` structure which indicates the appropriate values. However, the `1999.oct` data taken by the Caltech 40 meter interferometer did not have such a structure. When this occurs, default values of `locklow`=-4000 and `lockhi`=4000 are used. To change these default values, which, for clarity, are only used in the absence of a `locklow/lockhi` structure, the file `frameinterface.c` must be edited. The applicable portion of the code looks like:

```
if (staticdataL==NULL) {
GR_start_error("fget_ch()",rcsid,__FILE__,__LINE__);
GR_report_error("Unable to locate \"locklo/lockhi\" history structure in FRAME\n");
GR_report_error("It appears that there is no lock range information in these frames!\n");
GR_report_error("Defaulting to locklo=-4,000 lockhi=4,000\n");
GR_end_error();
fgetoutput->locklow=locklow=-4000;
fgetoutput->lockhi=lockhi=4000;
}
else {
/* pointer to the array containing the low/high values */
frvectlohi = (struct FrVect*) staticdataL->data;

/* record the low/high values internally, and return them to the user also */
fgetoutput->locklow=locklow=frvectlohi->dataS[0];
fgetoutput->lockhi=lockhi=frvectlohi->dataS[1];
}
```

To change the defalut `locklow/lockhi` values, the user need only change the lines in which `fgetoutput->locklow` and `fgetoutput->lockhi` are set to -4000 and 4000, respectively, above.

2.7 Function: `plot_frame`

The function `plot_frame` reads channel data and sends it to `xmgrace`, the successor to the `xmgr`, which computes an FFT and plots the FFT magnitude as a function of frequency. FFT's from multiple time intervals are plotted in succession, yielding an animated display.

2.7.1 Command Line Usage

The command line usage for `plot_frame` reads:

```
Usage: plot_frame [options] [Frame files]
Options:
```

```
-f<string>    Sets configuration file name to <string>.  This file
               must contain a list of the channels to be plotted,
               one per line, placed in decreasing order of sample
               rates.
-h            Prints this list
-n<double>    Sets number of seconds to be plotted at a time to <double>
-p            Toggles "print files" mode, in which commands are
               printed to .agr files that may be piped into xmgrace
               with cat *.agr | xmgrace -pipe instead of plotting them.
               Name of .agr files is N.agr where N is the starting
               GPS time rounded down to an integer.
               (DEFAULT = OFF)
-r<double>    Sets Frame sample rate to <double>
               Used only to relate number of points obtained to value
               provdied by -n switch...if indicated sample rate is wrong
               it is automatically corrected but not until after number
               of points to be obtained is determined...thus, actual
               number of seconds of data returned will not be correct
               (DEFAULT = 16384)
-s<integer>   Sets number of seconds to sleep between plots to <integer>
               (DEFAULT = 0)
```

If multiple channels are indicated in the `-f` configuration file, they are plotted simultaneously, arranged in rows, by `xmgrace`, with the first channel in the top plot, and so on. In addition, it is important to note that, unlike many other GWCLASS functions, `plot_frame` does not use environment variables to control its input. Rather, the Frames to be plotted **MUST** be provided on the command line.

2.8 Function: `print_freq`

The function `print_freq` prints to an ASCII text file a collection of ordered pairs giving the frequency of a specified pulsar as a function of time. The interval during which the frequency is computed and the spacing between points are specified by the user on the command line. Determination of the pulsar frequency is accomplished through use of output from tempo. See the description in section 2.1.1 for more information.

2.8.1 Command Line Usage

The command line usage for `print_freq` reads:

```
Usage: print_freq -[gmu] <tmin> <tstep> <tmax>

       -g Times in GPS seconds
       -m Times in MJD (Modified Julian Date)
       -u Times in UTC (Format MM/DD/YYYY-HH:MM:SS)
          (tstep still in seconds)
```

It is important to note that while most of the GWCLASS functions look for tempo output in files whose names contain the name of the target pulsar, such as <pulsar name>_polyco.dat, `print_freq` looks for the same output in a file with the precise name “polyco.dat”. For this reason, the pulsar name is not sent to `print_freq`. The user merely needs to make certain that “polyco.dat” contains tempo output for the appropriate pulsar.

2.8.2 Method

Though `print_freq` takes times in GPS, MJD, and UTC, use of the tempo files requires times to be expressed in MJD, so the appropriate conversion is made if necessary. GWCLASS uses the routine `get_coeffs()` to read the tempo output files, referred to as polyco files in comments contained within the code. The `frequency()` routine, which returns the pulsar's frequency, initially calls `get_coeffs()` to obtain a new set of polynomial expansion coefficients and its corresponding range of validity whenever the input time leaves the range of the previous set.

2.9 Function: `read_data`

The function `read_data` obtains data from an input stream of Frame files and prints both the raw data, as well as a power spectrum of that data, to ASCII text files. Raw data is printed to the file <channel name>.dat as ordered pairs depicting the channel value as a function of GPS time. The power spectrum of the data is printed to the file <channel name>_pow.dat as ordered pairs depicting the power as a function of frequency, with the latter expressed in Hertz. The FFT used when computing this power spectrum may be windowed with a window function, which is provided with a command line option.

2.9.1 Command Line Usage

The command line usage for `read_data` reads:

```
Usage: read_data [options]
  -c<string>             Sets channel to be read to <string>
                           (DEFAULT = IFO_DMRO)\n");
  -h                     Prints this help\n");
  -H                     Prints help info for environment variables\n");
  -l                     Turns on locked section option so that\n");
                           only data from locked sections is returned\n");
                           (DEFAULT = OFF)\n");
  -n<integer>            Sets number of points to be obtained to <integer>\n");
                           (DEFAULT = 16384)\n");
  -w<string>             Sets FFT window type to <string>\n");
                           Presently only HANNING and NONE are supported\n");
                           (DEFAULT = NONE)\n");
```

Input is achieved through the same input environment variables used by `clean_signal` and described in section 2.1.2. Note that only the discussion of input variables is relevant here, since `read_data` doesn't produce any output Frames. In addition, unlike `clean_signal`, `read_data` only reads one chunk of data, so only the first n points within the input stream are read.

2.10 Function: `utc2mjd`

The function `utc2mjd` converts an input time in UTC calendar format to MJD (Modified Julian Date).

2.10.1 Command Line Usage

The command line usage for `utc2mjd` reads:

```
Usage: utc2mjd [utctime]
       [utctime] has format is MM/DD/YYYY-HH:MM:SS
```

2.10.2 Method

First, the input string is parsed to obtain the number of days since a reference date, currently December 31, 1995, which has MJD 50083. That number of days, and a fraction corresponding to the fraction of a day equivalent to `HH:MM:SS`, is then added to 50083. It should be noted that the routine `gps_mjd()`, which is called by `gps2mjd` to convert GPS seconds to MJD simply converts GPS to UTC with the GRASP routine `gtime` and calls `utc_mjd()`, the same routine called by `utc2mjd`.

Chapter 3

GWCLASS Utility Scripts and Auxiliary Files

A few utility scripts and auxiliary files are included with the GWCLASS distribution to aid in its use. Their usage and purpose are described below.

3.1 Utility Scripts

The GWCLASS utility scripts are contained within the `GWCLASS_dir/utility_scripts` directory. The first line of each script must be edited to point to the location of the local `perl` or `bash` installation before use.

Many of these scripts are specific to certain types of file names. For this reason, the user may find it necessary to edit these scripts, or even write new ones. This is a reflection of the fact that the scripts contained within the GWCLASS package are not necessary for general GWCLASS usage, rather they are included primarily as an example of the sort of utilities that may prove useful.

3.2 Script: `e2d`

The script `e2d` is a simple script which reads, as input, an output file produced by the Tempo timing code in which scientific notation is written with an "E" and produces a new file, named `<input file>.out`, which is identical except each such "E" is changed to a "D." The necessity of this script relates to the manner in which the GWCLASS binaries `clean_signal` and `gwsearch` read the Tempo files. For more information, see section 2.2.3 above.

The usage for `e2d` is:

```
Usage: e2d <input file>
```

3.3 Script: `make_tar_list`

The script `make_tar_list` was written to aid in the processing of data from a remote server. Much of the data stored on CACR's HPSS archive consists of one-second Frame files collected into tar archives. Since `clean_signal` only processes data from locked sections, though, it is advantageous to avoid obtaining and unpacking tar files whose Frame files do not contain sufficiently long locked sections. As a result, it is desirable to compare a catalog of locked sections with a listing of those tar archives stored on HPSS to obtain a list of those archives that must be obtained for optimal processing. The script `mak_tar_list` accomplishes this for tar archives of a particular format.

The usage of `make_tar_list` is as follows:

```
Usage: make_tar_list <locked catalog file> <listing file>
```

The locked catalog file should contain a list of locked sections, having the same format as the output files produced by the binary `lockcat`, which are to be analyzed. The listing should contain a list of tar archives stored on a remote server. Each tar archive is assumed to contain 600 seconds of data and have name with the format `C1-<Start GPS Time>.F.n600.tar`. The output file produced by `make_tar_list`, bearing the name `tarlist.txt`, contains those tar files corresponding to the locked sections listed in `<locked catalog file>`.

3.4 Script: `run_cleaning`

The script `run_cleaning` sets the appropriate environment variables and calls `clean_signal` with a number of command line options. It is suggested that any user of `clean_signal` either modify `run_cleaning` or write another script which sets appropriate environment variables prior to calling `clean_signal`.

3.5 Script: `process_listing`

The script `process_listing` was written during the course of obtaining lists of remotely stored tar archives of the type needed by the `GRASP_FTPFILE` environment variable (see section 2.1.2 above). It takes as input the output of a call to `ls -l` and produces, as output, a file named `<input file>`.out, which contains the names of the files listed in the input directory, with one per line. This utility was needed because, during the author's work, calls to `ls` within `hsi` produced three columns of output, with files arranged in order such that the first file of the second column followed the last file of the first column. These files would be read in the wrong order by GWCLASS binaries, causing serious problems. Thus, `ls -l` was called to ensure that file names appeared in one column, and thus would be read in the proper order, and `process_listing` was called to remove all extraneous information contained within the listing, leaving only the names of the tar archives to be obtained.

The usage for `process_listing` is:

```
Usage: process_listing <list>
```

As a final note, it is important to note that `process_listing` assumes that the names of files within the listing are of the form `C1*`.

3.6 Auxiliary Files

Auxiliary files contained within the GWCLASS distribution are contained in the directory `GWCLASS_dir/aux_files`. These files are examples of the sort of input files required by GWCLASS binaries, and are listed in table 3.1.

File	Description
`1937+21.param`	Sample parameter file, needed by `gwsearch`, as well as `clean_signal` if the signal injection feature is used, containing right ascension and declination of PSR1937+21
`1937+21_polyco.dat`	Sample output file from the Tempo timing code which was produced for PSR1937+21 using a parameter file included with the Tempo distribution. The data within this file corresponds to observations made at a Princeton University site, rather than any of the currently operating gravitational wave interferometers.
`catalog_256.txt`	Sample catalog of locked sections from a portion of the `1999.oct` run of the Caltech 40 meter interferometer from which all locked sections of length < 256 seconds have been removed.
`corr.config`	Sample configuration file for `clean_signal`, containing a list of channels to be removed.
`listing.sample`	Sample file for `GRASP_FTPFILE` environment variable, containing a list of tar archives to be obtained.
`ncftp.config`	Sample `ncftp` configuration file required for using `ncftp` for remote input or output
`pulsars.txt`	Sample `-i` input file for `clean_signal` which contains information for signal injection

Table 3.1: Auxiliary Files included with GWCLASS

Bibliography

[1] B Allen. *GRASP Analysis Code.* `http://www.lsc-group.phys.uwm.edu/~ballen/grasp-distribution`.

[2] B Allen, W Hua, and A Ottewill. Automatic cross-talk removal from multi-channel data. *LIGO Internal Document*, 1999. `LIGO-P99000`.

[3] W Anderson, J Whelan, P Brady, J Creighton, D Chin, and K Riles. Beam pattern response functions and times of arrival for earthbound interferometers. *Maple Worksheet*, 2001. `http://phys.utb.edu/UTBRG/activities/papers/#UTBRG-2001-01`.