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

Document Type	**LIGO-T952008-00 - R**	3/9/94

A Tutorial For the Fast Fourier Transform Interferometer Simulator

Brett Bochner
Yaron Hefetz

Distribution of this draft:

This is an internal working note
of the LIGO Project..

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (818) 395-2129
Fax (818) 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/

Contents

Chapter 1 Introduction to the Program and its Uses

This manual is intended as a guide to the program that has been written to simulate the steady-state fields of the core region of the LIGO interferometer. This simulated region consists of a recycled, Michelson interferometer with a Fabry-Perot cavity in each of the Michelson arms.

The earliest version of this simulation program was written by VIRGO scientists Jean-Yves Vinet, Patrice Hello, Catherine N. Man, and Alain Brillet. The code was modified at LIGO by Yaron Hefetz and Partha Saha to increase rapidity of convergence, and the algorithms for the simulation of the full-LIGO interferometer were written by Brett Bochner. A schematic picture of the simulated physical system appears in Fig.1, "Diagram of a Full-LIGO Interferometer", on page 4.

Perhaps the best and simplest introduction to the program can be given in terms of the physical assumptions of the model, the types of simulations we can perform with the program, and the basic locations of the files that can be copied and utilized by prospective users.

The Physical Assumptions and Parameters of the Simulation

- The LIGO simulator is a numerical algorithm performed on an NxN grid (typically, N=128). Each electric field, mirror, and propagation operator in the program is defined by the N^2 pixels which occupy its grid.
- A steady-state condition is assumed for all aspects of the program. In particular, the fields are considered relaxed when they do not change significantly after a round-trip through the interferometer. The mirrors are assumed fixed for long periods of time, and there are no transient effects.
- The laser is ideal. The excitation of the interferometer is supplied by a perfect Hermite-Gaussian TEM_{00} beam (supplied by a hypothetical CW Argon-Ion laser). The beam possesses no noise of any kind, other than pixelization and computer round-off errors. There are no restrictions on the choice of frequency or frequencies which can be used to illuminate the interferometer.
- For propagation over long distances ($>>\lambda$), the Paraxial Approximation is assumed. This approximation allows us to perform these propagations within a convenient Fourier Transform mathematical framework. The primitive engine

of the simulation is a generic Fast Fourier Transform algorithm, and the LIGO simulation program will often be referred to as the “fft program” in this manual.

- For propagation over short distances ($<<\lambda$) to an uneven surface, each pixel of the propagated field is multiplied by a phase factor corresponding to the distance that pixel must travel.
- The interferometer is resonant. All laser parameters (such as waist, radius of curvature) are designed to achieve coupling of the excitation field into the entire interferometer. The distances between the optical elements are adjusted to achieve a specific definition of a “resonant" condition.

The Physical Variety of Simulations

The basic types of runs that can performed may be placed under one or several of these general headings:

- Carrier Runs : The entire interferometer is resonant. Power in the Fabry-Perot arms is maximized, and power in the recycling cavity is the maximum possible given the FP arm resonances.
- Sideband/Subcarrier Runs : Only the recycling cavity is resonant. Recycling cavity power is at the maximum possible value. The FP arm cavities are far-from-resonant (they contain close to the anti-resonant level of power).
- Unbalanced Interferometer Runs : A macroscopic length imbalance may be given to the two Michelson arms. The exact lengths are then recalculated (on the scale of a wavelength) to bring the interferometer back to the desired level of resonance.
- Misaligned Mirror runs : Simulation runs may be performed with tilted mirrors and with mirrors displaced transversely with respect to the incoming laser beam. These misalignments are not removed or altered by the running program.
- Imperfect Mirror Runs : Imperfect surfaces and substrates can be applied to the various mirrors of the program to simulate degradation of interferometer response due to imperfectly constructed mirrors.

An “encyclopedic” listing of runs performed with the full-LIGO simulation program is included in Appendix C. A more descriptive discussion of the physical results, written by Yaron Hefetz, is available on **kepler.mit.edu** in the publisher

file:

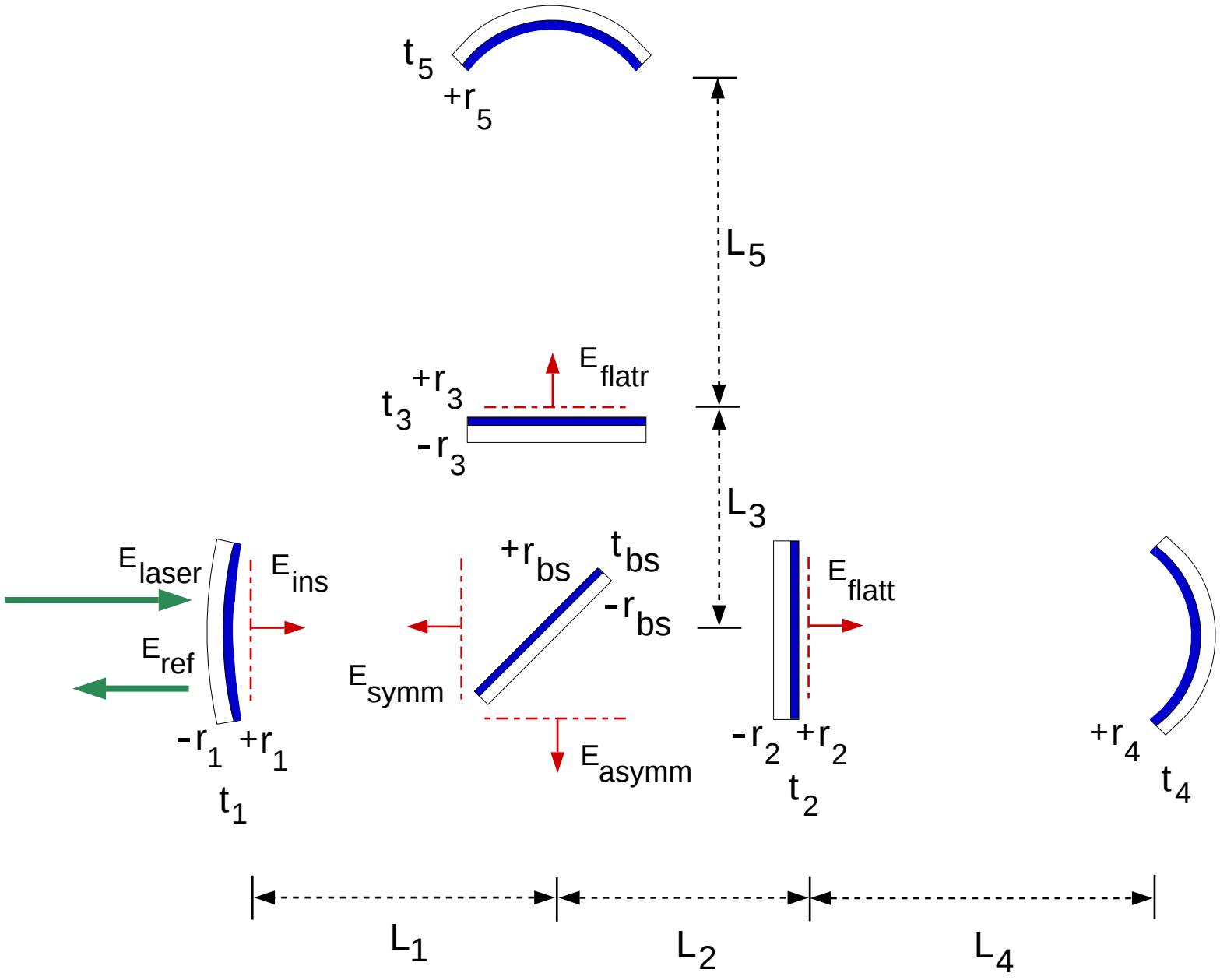
~brett/fft_documentation/fft_user_files/LIGO_run_descriptions.pub

Some Lotus123 files containing the summarized output numbers from some of the runs are also available, in the subdirectory:

~brett/fft_documentation/fft_user_files/Lotus123_FFT_results

(Contact **yaron@tycho.mit.edu** for an explanation of these data files.)

Fig.1 : Diagram of a Full-LIGO Interferometer



Physical Input Parameters for Program :

1) Wavelength of laser.
2) Spatial mode shape of laser beam (TEM_{00}).
3) Macroscopic Distances L_1- L_5 .
4) Radii of curvature of FP back mirrors.
5) Reflectivities of mirrors.
6) Losses in mirrors.
7) Diaphragms of mirrors.
8) Mirror tilts.
9) Mirror displacements (transverse to beam).
10) Mirror base thicknesses (in wavelengths).
11) Mirror surface & substrate imperfections.
12) Transverse offset of laser beam.
13) Size of square calculational window.

Physical Parameters Calculated By Program :

1) Waist size of laser beam.
2) Radius of curvature of recycling mirror (matched to E_{ins}).
3) Microscopic adjustments to L_2 ,L_3 ,L_4 ,L_5 .
4) Steady-state fields E_{ins}, E_{ref} , E_{flatt}, E_{flatr} , E_{symm}, E_{asymm} .

The Computer File Structure

A typical usage of the simulation program consists of three operational stages: pre-processing, fft program execution, and post-processing.

Pre-processing involves steps such as creating deformation files for the interferometer's mirrors, editing the appropriate input files for the current run, etc. These steps are generally performed on both the Sun and the Cray, and this procedure is more dependent on the user's particular choices than are code execution and post-processing.

The actual execution of the fft code occurs on the Cray supercomputer. This is where the main program and input files reside, and this is where the majority of computational time is spent.

Once this execution has been completed, the output files from the simulation are transported down to the Sun computer, where all the post-processing (such as modal decomposition of the relaxed fields) is performed.

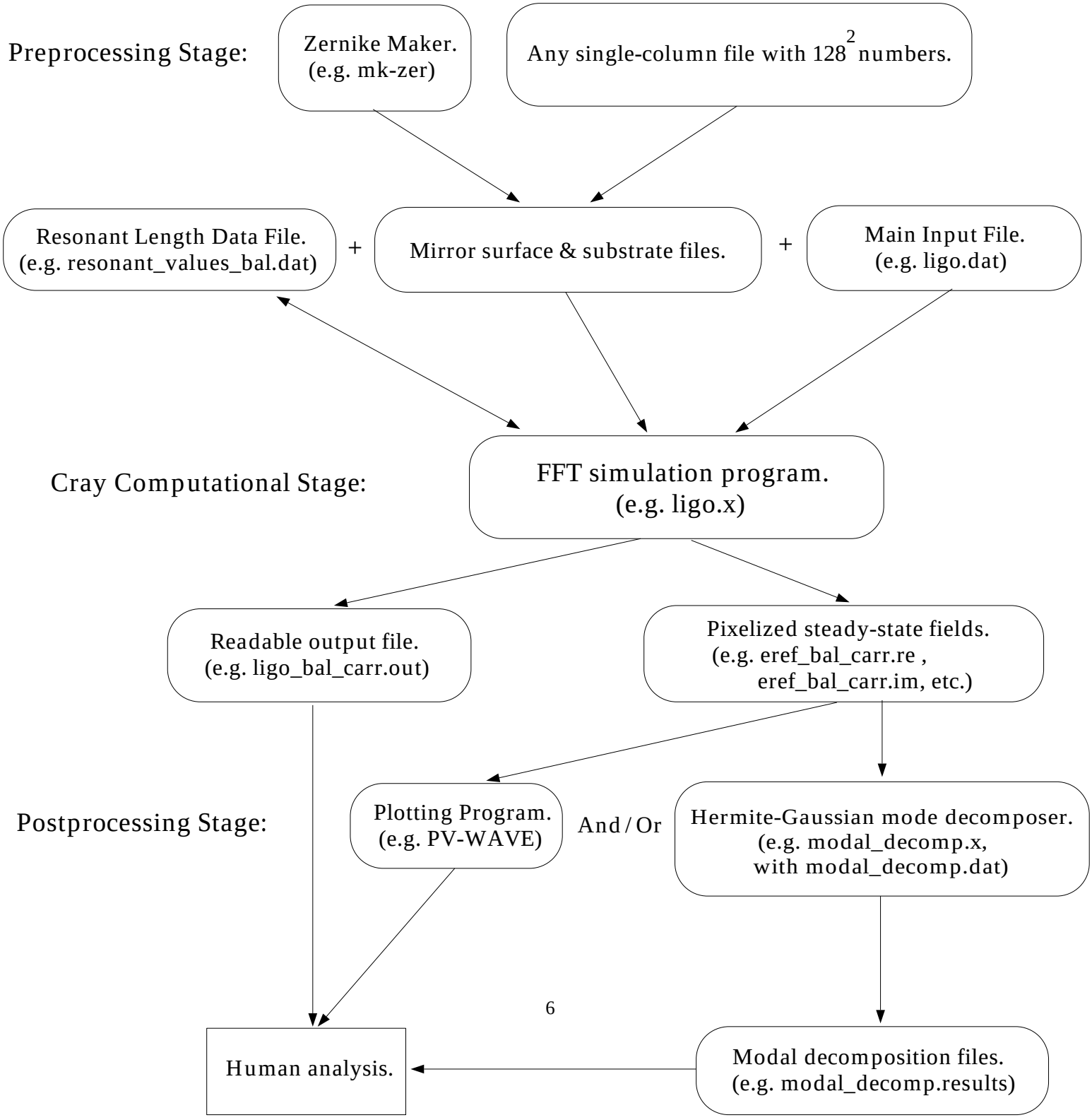
As will be mentioned often, all of the files referred to in this document are available to prospective users of the fft program. Copies of these files are located on the Sun computer **kepler.mit.edu**, within the directory:
~brett/fft_documentation/fft_user_files

This includes all of the files which you will need to bring to the Cray for program execution. Copies of these files are located within the Sun "fft_user_files" directory, in the subdirectory:
~brett/fft_documentation/fft_user_files/cray_files

For a schematic view of the interrelationship between all the files necessary for a start-to-finish run of the simulation process, we have provided Fig.2, "File Hierarchy for the Interferometer Simulation Program", on page 6.

The substance of this manual officially begins in the next chapter with a "How-To Guide" written in order to make a first execution of this program as painless as possible to the new user. If you have any questions or complaints (or even good suggestions) about the program or this manual, please address your comments to one of us:
brett@taliesin.mit.edu (Brett Bochner)
yaron@tycho.mit.edu (Yaron Hefetz)

Fig.2 : File Hierarchy for the Interferometer Simulation Program



Chapter 2 How to Use the Program: From Quick & Dirty to Elegance and Style

Section 1 Performing an Elementary Run — A How-To Guide

All of the commands below are demonstrated in an accompanying sample login session located on **kepler.mit.edu**, in the file:
~brett/fft_documentation/fft_user_files/how_to_guide_sample_session

1. NOTE: All the Cray files that you will need to copy to your Cray account are located on the computer **kepler.mit.edu**, in the subdirectory: **~brett/fft_documentation/fft_user_files/cray_files** .
2. Log on to *your personal* Cray account, and go to the subdirectory where you will perform your LIGO simulation runs. (If you do not have a Cray account, or have any problems remote-logging on to the Cray, contact: **yaron@tycho.mit.edu**.)
3. Copy the following files from the "cray_files" subdirectory into your Cray running directory:
   - **ligo.x**
   - **ligo_bal_carr.dat**
   - **ligo_run**
   - **recyc_perf.dat**
   - **online_inp_perf.dat**
   - **offline_inp_perf.dat**
   - **online_back_perf.dat**
   - **offline_back_perf.dat**
4. Type the command: **cp ligo_bal_carr.dat ligo.dat**
5. Prepare the batch submission file, **ligo_run**, by editing this file and replacing the line:

cd ligo

with the line:

cd <*your_directory*>

where <*your_directory*> is the subdirectory where all of your Cray files are kept, and from where you will run the simulation.

6. Run the simulation by typing: **qsub –lT 3550 –lM 10MW ligo_run** (The only number 1 in this command is in the number 10; The others are lower case L's. The dashes are single minus signs.)
7. To check if the run has started and is still going, type the command:
   ps –u *<your_name>*
   where "<*your_name*>" means *your Cray username.*
8. If it is still running, the computer will respond with something like:

```
  PID TTY TIME COMMAND
39287 p015  0:00 quotamon
48258   -   0:00 csh
39283 p015  0:01 csh
48284   -   1:42 ligo.x
49150 p015  0:00 ps
48266   -   0:00 quotamon
```

This means that **ligo.x** has been running for 1 minute and 42 seconds (in CPU time).

9. If the program has completed running, the computer will respond with something like:

```
PID TTY TIME COMMAND
50312 p015  0:00 ps
39287 p015  0:00 quotamon
39283 p015  0:01 csh
```

Observe that the **ligo.x** run has disappeared from this message.

10. When the run has finished, type: **ls –l ligo_bal_carr.out** . Make sure that the date & time given for this file matches the date & time when your run just ended. If it does not, then you must:
    a. Type the command:**rm ligo_bal_carr.out**
    b. Return to step 4 above and proceed again from there, to get a new output file.

11. Type: **more ligo_bal_carr.out** , and briefly examine the file on your screen. Do not worry about understanding it yet.
12. You may now log out of your Cray account.
13. Log on to your account on the Sun.
14. Create a subdirectory called **fft_runs** .
15. On the computer **kepler.mit.edu**, there is a subdirectory named:
    ~brett/fft_documentation/fft_user_files
    Copy that directory and *everything* inside of it into your Sun directory **fft_runs**. Notice that you are not just copying files, you are copying subdirectories of files, which have a fixed hierarchy. This task is accomplished with the following command:
    cp –rR ~brett/fft_documentation/fft_user_files <*your_path*>/fft_runs
    where <*your_path*> refers to the location of your "fft_runs" subdirectory.
16. Move to your new decomposing subdirectory by typing:
    cd <*your_path*>/fft_runs/fft_user_files/mode_decomposition
17. Use the FTP utility to copy the following files from your Cray directory to your current Sun subdirectory. (For a sample session with the FTP command, see the subsection of chapter 2 entitled, "Postprocessing for the Simulated Electric Fields : Getting the Files from the Cray".)

    - **ligo_bal_carr.out**
    - **eins_bal_carr.re**
    - **eins_bal_carr.im**
    - **eflatt_bal_carr.re**
    - **eflatt_bal_carr.im**
    - **eflatr_bal_carr.re**
    - **eflatr_bal_carr.im**
    - **esymm_bal_carr.re**
    - **esymm_bal_carr.im**
    - **easymm_bal_carr.re**
    - **easymm_bal_carr.im**
    - **eref_bal_carr.re**
    - **eref_bal_carr.im**

18. Rename and print out the human-readable output file by typing the commands:

    mv ligo_bal_carr.out ligo_first_run.out
    enscript –2 –r –G –h ligo_first_run.out

19. Clear the area with the commands: **rm *results** , and: **rm *numbers** .
20. Copy the file **decomp_bal_carr** to a new file, **decomp_active** , which you will use for decomposing all of the output fields at once.
21. Edit **decomp_active** : in this file, change *every* occurance of the phrase "**CHANGE_THIS_PART**" to the phrase "**FIRST_RUN**".
22. Decompose your electric fields by typing the command: **decomp_active**
23. Wait 10 minutes or so, until your computer prompt returns. Talk to a friend. Get a snack.
24. Waste paper (this time) by printing out all your results with the commands:

enscript –2 –r –G –h *results
and:

enscript –2 –r –G –h *numbers
Collect your results from the printer and look them over.

25. Create a unique subdirectory for these results by typing:
    mkdir first_run_output
26. Store these output files safely with the commands:

    mv ligo_first_run.out first_run_output
    mv *results first_run_output
    mv *numbers first_run_output
    mv e*re first_run_output
    mv e*im first_run_output

27. Congratulations, you are done with your very first run!
28. If you have any questions about the simulation program or this manual, feel free to contact: **brett@taliesin.mit.edu** (Brett Bochner) or **yaron@tycho.mit.edu** (Yaron Hefetz).

Section 2 Understanding the Mechanics of the Run

There are three elements one absolutely needs access to in order to perform a run with the fft numerical relaxation program:

1. A compiled, executable version of the Fortran source code.
2. Proper input files; this includes a general input file containing parameters for the program to read, and several descriptor files for the mirror surfaces.
3. Access to the MIT Cray, or an equivalent supercomputer capable of executing vectorized code. (Getting and/or logging on to a Cray account is not described in this manual, and it is essentially the responsibility of the user of the simulation program.)

The Executable Code, ligo.x

- The executable code, located on **kepler.mit.edu** in the subdirectory: **~brett/fft_documentation/fft_user_files/cray_files** , is called **ligo.x**. This file is necessary (but not sufficient) to perform a simulation run.
- To avoid difficulties, *do not* attempt to execute the code until you have read through the section below entitled, "Running the Program".

Commentary:

The latest version of the Fortran-based source code is currently called **ligo.f** . The code will be under modification in the near future, but this version will not be affected by those changes. This version performs a full simulation of a complete LIGO interferometer for various laser frequencies and resonance conditions. The compilation command that has been used (on the Cray) to create the executable code is:

f77 –Zv –Wf' –dp' –o ligo.x ligo.f

"f77" is the command to invoke the Fortran-77 compiler. "-Zv" instructs the compiler to recognize vectorized code. " ' –dp' " turns off Cray double-precision (Cray single-precision is equivalent to normal double-precision). " –o ligo.x ligo.f" directs the compiler to write the executable version of **ligo.f** to a file named **ligo.x** .

Essential Input Files on the Cray

The Main Input File, ligo.dat

This file contains the fundamental parameters read in by the ligo fft program. The active main input file must ALWAYS be named "**ligo.dat**" when you run the program. A copy of this input file for a "perfect" ligo interferometer is included in Appendix A of this document. We will refer to it often throughout this manual. There are certain data in this file which you must ALWAYS be aware of and choose carefully before any run:

a. Maximum Iterations : On the next line after the comment, **"Maximum # of relaxation iter's . . ."**, there is an integer at the beginning of that line which represents the maximum number of steps the program will use to converge all of the electric fields, before it gives up. In the example data file in the appendix, the number is 1200. This is a *large number* of relaxation steps. A good rule of thumb is this: One iteration is a tiny bit more than one Cray Cpu second. One Cray Cpu hour costs $50 cash money. Therefore, 1800 iterations, if they are all performed, will cost slightly more than $25 (and will take about 30 minutes of "real" time.).

b. Running from Scratch : On the next line after, "**Bring exact lengths to resonance (1) or** . . . " , there will be an integer; a 1 or a 0. For a beginning run, it is important to choose "1" . Using the "0" option requires the presence of a data file that has been created (with special requirements, to be explained later) by a previous "option 1" run.

c. Name of Main Output File : On the next line after, "**Name of main output file:**", is the filename of the readable-format output file that will describe the success of the run and some significant numbers pertaining to the relaxed fields themselves. You must keep track of this data file entry for each run, to avoid mixing up the output filenames. (Methods of keeping track of filenames is provided in the manual section on "Run Templates".) Note that any previously existing output file with the same name will be *overwritten* by the current run.

The Mirror Descriptor Files

Types of mirror descriptions in the simulation:

Recycling Mirror — Reflective surface & Substrate
On-line, off-line input mirrors — Reflective surfaces & Substrates

On-line, off-line back (4 km distant) mirrors — Reflective Surfaces only
Beam-splitter — no physical map, just a numerical attenuation of the beam

It is *necessary* to have on hand all five distinct mirror surface descriptor files during each and every run.

Mirror substrate descriptor files are *optional*. The recycling & arm cavity input mirrors may be given files for non-ideal substrates. A "+" sign in place of a substrate filename in **ligo.dat** indicates that you are declining this option, and using an ideally uniform substrate.

Five files for "perfect" (undeformed) mirror *surfaces* exist in the "cray_files" subdirectory, and are used for all runs with perfect mirrors. They are called:

- **recyc_perf.dat**
- **online_inp_perf.dat**
- **offline_inp_perf.dat**
- **online_back_perf.dat**
- **offline_back_perf.dat**

These five mirror surface files, in particular, are the ones used in the sample **ligo.dat** file in Appendix A. They are specified in the region after the line:
"**Names of the files containing variations in the mirrors** . . . "
The five lines which currently have filenames in them must always contain the names of your five mirror surface files.

File Format : Each surface and substrate file contains a single column of data. There are 128*128 = 16,384 lines of data, one entry (a real number) per line. Each data entry represents a single pixel in the 2-D array. Each entry is a physical distance (assumed in meters), which is converted to a phase shift for light at the given laser frequency. Therefore, a perfectly smooth surface or substrate file will have 16,384 lines, each line containing the real number zero. For general surfaces, each numerical entry is written in the Fortan exponential format, e.g. "**-1.23433e-09**".

Also note:

- Each mirror filename must be 25 characters long or less.
- Each mirror file must be in the same subdirectory as the running program.

Running The Program

Before running is possible, it is necessary to obtain an account on a supercomputer. Running the simulation on a slower computer (say, a Sun SPARCstation 10) could take five or more hours to run. In addition, segments of the code are written in "vectorized" fashion, so that only a vectorizable computer will run it free of error. Furthermore, the code as a whole is *not currently compatible* with any system other than the Cray. Interested users may have to spend some time converting this program to run on their local supercomputer.

For running on the MIT Cray (once you have obtained the necessary input files described in the previous sections):

1. In Batch Mode (*preferred*) :

   a. Copy our sample script file, **ligo_run**, from the Sun "cray_files" subdirectory into your own Cray working directory.
   b. Read steps 5 & 6 in Chapter 2, Section 1 ("Performing an Elementary Run : A How-To Guide"), and edit **ligo_run** according to those directions.
   c. Type the command:

      qsub –lT 3550 –lM 10MW ligo_run

      Regarding the run-time limit:

      - In this qsub command, 3550 is the maximum number of CPU seconds that you are alloting to the Cray to run your program.
      - A larger number than 3600 here can be *very bad* : that would put your job on the infinite-time queue, and you could lose all of your Cray money if your run gets hung up for some reason and fails to terminate until it runs out of money.
      - A smaller number than 3550 can be *somewhat bad* : your job may run on a faster queue, *but* if the program does not stop by the CPU time limit, it will be killed summarily, and all output from this run will be lost. Always limit your job run-times with **ligo.dat** iteration limits, not with CPU time limits.

   d. To check on your job's progress, follow steps 7–9 in the "How-To Guide" in Sec.1 of this chapter.

e. Active, unwanted batch jobs may be terminated with the command:
kill –9 <PID number>
where you can obtain the job's PID number by following step 7 in the "How-To Guide".

2. Interactively (*for short runs, and few of them*) :

a. Type the command: **ligo.x**.
b. Your Cray prompt will return when the run is finished.
Beware :

- You are only allowed a *cumulative total* of 15 CPU minutes of interactive running per login. When this limit is exceeded, the Cray will kill your job and log you out automatically.

A Brief Understanding of the Output

The human-readable main output file: **ligo_bal_carr.out**

The output files for the simulated electric fields:

- **eins_bal_carr.re**
- **eins_bal_carr.im**
- **eflatt_bal_carr.re**
- **eflatt_bal_carr.im**
- **eflatr_bal_carr.re**
- **eflatr_bal_carr.im**
- **esymm_bal_carr.re**
- **esymm_bal_carr.im**
- **easymm_bal_carr.re**
- **easymm_bal_carr.im**
- **eref_bal_carr.re**
- **eref_bal_carr.im**

Be aware that the output filenames produced by the simulation program *do not* always match the above listing; output filenames are actually options which are set in **ligo.dat**. As an example, however, the **ligo.dat** file presented in Appendix A does produce output files which match the above list.

The Main Output File (e.g., ligo_bal_carr.out)

This file is meant to be human-readable without any processing. The file essentially exists in four segments:

1. An echo of the input parameters: Having the input and primary output data in the same file is very useful for checking that no errors in the input information lead to incorrect interpretations of the results.
2. Mid-program calculations and run diagnostics: Ignore these for now.
3. Physical output numbers, such as:
   - Laser power (in all modes) at seven locations in the interferometer.
   - Contrast defect, "1–C".
   - Cavity gains and various "alphas" (measures of decreased power relative to a perfect, unperturbed interferometer).
4. CPU time usage for various parts of the program, and a quote for the entire run. Remember, budgeting time and computer money properly is essential!

The Simulated, Relaxed Electric Field Files

1. Each *complex* field is written out into two files: a ***.re** file for the real part of the field, and a ***.im** file for the imaginary part of the field. (Each of these output files consists of *real numbers only*.)
2. Six fields are written out — twelve E-field files in all, per run.
3. The six interferometer locations are (also see Fig. 1, page 4):

   eins = Next to the recycling mirror, *inside* the recycling cavity. (Field is propagating towards the beam-splitter.)
   eref = Next to the recycling mirror, *outside* the recycling cavity. (Reflecting back to the laser.)
   eflatt = Next to the on-line input mirror, *inside* the on-line arm cavity. (Propagating towards the back mirror, 4km distant.)
   eflatr = Next to the off-line input mirror, *inside* the off-line arm cavity. (Propagating towards the back mirror, 4km distant.)
   easymm = Next to the beam-splitter, being ejected from the asymmetric interferometer port.
   esymm = Next to the beam-splitter, being emitted from the symmetric interferometer port. (Propagating towards the recycling mirror.)

4. The total power in each of these fields is reported in the main output file (e.g. **ligo_bal_carr.out**). These values are relative to that of the excitation laser beam (called **phiin**); **phiin** is always a Hermite-Gaussian TEM_{00} with power normalized to unity.
5. Each of these electric field files in 128*128 = 16,384 lines long; too long to be conveniently readable.

The two methods we use to convert these files into useful information are graphical plotting and modal decomposition. Both forms of analysis are performed *on the Sun computer*, not on the Cray.

Postprocessing for the Simulated Electric Fields: Getting the Files from the Cray

We use FTP commands to transport the output files from the Cray to the Sun. The basic commands we use are:

ftp cray — (opens the session; user must then log in normally)
cd <*subdir*> — (in my account, <*subdir*> = **ligo**)
get <*filename*> — (downloads a single file to the Sun)
mget <*filenames*> — (downloads several files; asks yes/no for each before proceeding)

The following is a sample downloading session. Commands that you must type are in **bold**. Comments that we have added later are in *italics*. Prompts and commands that are user-dependent appear in <brackets>.

<your prompt> **ftp cray**
Connected to ED.MIT.EDU.
220 edtoo FTP server (Version 5.2 Fri Sep 7 14:09:58 CDT 1990) ready.
Name (cray:<default>): **<your Cray username>**
331 Password required for **<your username>**.
Password: **<your password>**
230 User <username> logged in.
ftp> **cd <~brett/ligo>** *{Change to your directory with the simulation files.}*
250 CWD command successful.

ftp> **get ligo_bal_carr.out** *{It is good to download the output file together with the decomposable files.}*
200 PORT command successful.
150 Opening ASCII mode data connection for ligo_bal_carr.out (5371 bytes).
226 Transfer complete.
local: ligo_bal_carr.out remote: ligo_bal_carr.out
5492 bytes received in 0.016 seconds (3.3e+02 Kbytes/s)
ftp> **mget e*.re** *{get all 6 of the real-part electric field files}*
mget easymm_bal_carr.re? **y** *{Answer "y" for each file you want to copy down. Answer "n" for each file you don't want to copy.}*
200 PORT command successful.
150 Opening ASCII mode data connection for easymm_bal_carr.re (360448 bytes).
226 Transfer complete.
local: easymm_bal_carr.re remote: easymm_bal_carr.re
376832 bytes received in 0.9 seconds (4.1e+02 Kbytes/s)
mget eins_bal_carr.re? **y**
{and so on, for the remaining real-part field files}
ftp> **mget e*.im** *{get all 6 of the imag-part electric field files}*
mget easymm_bal_carr.im? **y**
{and so on, for the remaining imaginary-part field files}
ftp> **quit**
<your prompt>

You are now ready to begin analyzing these files in qualitative and quantitative detail.

Postprocessing for the Simulated Electric Fields: Graphical Plotting

You may use any convenient graphical plotting package to view the output electric fields. We have found the graphics package PV-WAVE (by Visual Numerics International) to be particularly useful for reading in the data files and plotting them on Sun X-Windows. In the course of our work, we have written several PV-WAVE-based procedures which facilitate manipulations of the fields, both graphically and algebraically. Some of these procedures are available on **kepler.mit.edu**, in the subdirectory:

~brett/fft_documentation/fft_user_files/PVWAVE_plotting.
For further information about these and other available subroutines, send e-mail to:
brett@taliesin.mit.edu .

Postprocessing for the Simulated Electric Fields: Mode Decomposition

A more quantitative way to study the composition of the output fields is to project each field upon the complete, orthogonal basis set of Hermite-Gaussian transverse modes. These modes have two free parameters at each point: the "spot size", and the "radius of curvature". For each location in the interferometer where the relaxed electric fields are recorded, an approximate calculation will produce good values for these two parameters, which are necessary input for the mode decomposition program.

In the simulation, the interferometer is illuminated with a perfect TEM_{00} beam of known waist size and plane of focus. Therefore, the presence of higher-order HG transverse modes in the relaxed output fields is an indicator of deviations of the interferometer from perfect alignment and perfect optics.

All of the files relevant to this modal decomposition algorithm are available on **kepler.mit.edu**, in the subdirectory:
~brett/fft_documentation/fft_user_files/mode_decomposition

For a no-frills decomposition run, all you need are these files:

1. **modal_decomp.x** : the executable code for the decomposition
2. **modal_decomp.dat** : the decomposition main input file
3. *Two electric field files* : These files hold the real & imaginary parts of the fields you are decomposing. They must be named in the appropriate places in **modal_decomp.dat**. For the sample decomposition input file kept in the "mode_decomposition" subdirectory, the real & imaginary parts of the field that will be decomposed are contained in the files **esymm_bal_carr.re** and **esymm_bal_carr.im**.

To run the decomposition, you bring the appropriate files to your directory, and then at your prompt you type the command:

modal_decomp.x

The decomposition run should take a few minutes to finish. The output files from the decomposition run will be named:

1. **modal_decomp.results** : This is a listing of the breakdown of the complex field into the modes ennumerated in **modal_decomp.dat**. For each modal component, the amplitude, phase, real part, imaginary part, and squared amplitude (i.e. total power) of that component are written out. The total power in the field is also written out.
2. **modal_decomp.numbers** : This is a simplified version of the previous file. Only the total power in each modal component, and the total field power, are written out.

Further Notes:

- More useful decomposition runs, which decompose all fields in the interferometer simultaneously, are described in Ch. 2, Sec. 3, "Physically Interesting Runs and Run 'Templates'".
- You must rename your decomposition output files after each run, or they will be overwritten by the next run of the decomposer.
- You should also rename the electric field files, if you wish to keep them, so that they are not overwritten by the next fields you bring over from the Cray.

Upon examining the main decomposition input file, you will notice that it takes many parameters:

- the names of the fields to be decomposed
- the number of modes to decompose the field into
- (optional) horizontal or vertical offset of the center of each mode from the center of the decomposable field
- the wavelength of the light for construction of the modes (we always use the carrier wavelength, since it is close, and since the chosen sideband frequency may change)
- the physical size of the window of the decomposition (must match the value used in **ligo.dat**)
- The distance of the field *beyond* the plane containing the beam waist, i.e. the focal plane. (This value is negative for a field approaching its focal plane.)
- A listing of spatial indices and waist sizes for the construction of at least as many Hermite-Gaussian modes as will be used for the mode decomposition.

(If you have chosen to decompose into N modes, then the first N modes in this list will be used.)

Because we run the LIGO simulation with a set of basic physical dimensions which do not change from run to run, we have been able to write 24 general decomposition data files, which are appropriate for the six different fields written out by the simulation program, and the four general types of simulation runs we perform (carrier & sideband runs, balanced & unbalanced interferometer runs). Copies of these general data files are located in the "mode_decomposition" subdirectory, and they are named:

modal_decomp_<*suffix*>.eins_dat
modal_decomp_<*suffix*>.eref_dat
modal_decomp_<*suffix*>.eflatt_dat
modal_decomp_<*suffix*>.eflatr_dat
modal_decomp_<*suffix*>.easymm_dat
modal_decomp_<*suffix*>.esymm_dat

Where "<*suffix*>" may be "bal_carr", "bal_sb", "unbal_carr", or "unbal_sb". (These suffixes are explained in the next section, in the subsection "Run and Decomposition Templates".)

If you do not know which decomposition data file belongs to which of the six interferometer electric fields, see Fig.1, "Diagram of a Full-LIGO Interferometer", on page 4.

To decompose with any of the above data files, follow these steps:

1. Copy the data file you wish to use into **modal_decomp.dat**.
2. Look at the text of this data file, and make sure that the electric field files it asks for (the files that you are decomposing!) are present in your directory.
3. Type the command: **modal_decomp.x**
4. Rename **modal_decomp.results** and **modal_decomp.numbers** so that their filenames reflect which field they represent, and what kind of simulation run they come from.

Note that you must have all 24 of these decomposition data files in your directory to use the "run templates" that are described in the next section.

Section 3 Physically Interesting Runs and Run "Templates"

There are many parameters which you may change in **ligo.dat**, and also in the postprocessing decomposition runs. Nevertheless, some runs are inherently more interesting than others. Input variables such as macroscopic arm cavity lengths, mirror reflectivities and losses, etc., have originally been set according to the specifications of the real LIGO interferometer, and changing these parameters in the input files may be of negligible interest. Some other options of the simulation — such as running with a carrier or sideband frequency — may be of considerable interest. We have therefore created several run "templates", both for the actual simulation and the post-processing steps, which greatly reduce the number of editing operations necessary to perform interesting sets of runs.

Before we can discuss these templates, however, we must discuss the two main sets of options of the simulation program: carrier vs. sideband runs, and balanced vs. unbalanced Michelson cavity runs.

Field Relaxation vs. Field Resonance

The ligo simulation program must perform not one, but two operations simulataneously: it must relax the electric fields in the entire interferometer, and it must fine-tune the lengths of the cavity arms to ensure that the entire system maintains a resonant condition in this steady-state. We may define these two conditions as follows:

- **Steady-State Condition**: We speak of the interferometer as being in a steady-state when NONE of the fields monitored in the system change by more than a pre-specified, acceptable amount during the course of one additional round-trip propagation of all fields (including the laser excitation field) through the interferometer.
- **Resonant Condition** : This is a more complicated issue, and there are several ways to define resonance, all of which are hopefully (but not necessarily always) equivalent:

    a. Resonance is obtained when a specific field or fields (in this case, the fields in the FP arm cavities) have the maximum power that may be obtained via small changes in the mirror-to-mirror distances.

b. Resonance is obtained when all external excitation fields entering a cavity are exactly in phase with the internal-cavity fields that they are exciting.
c. Resonance is obtained when an excitation beam reflected from a cavity will experience a phase shift of exactly π.
d. Resonance is obtained by minimizing an "error signal" derived from the light that escapes from the interferometer. This error signal is generated by mixing several frequencies of light which have been relaxed together in the interferometer.

Among these choices, and perhaps others, the fft simulation program obtains resonance via method (b). In a very limited set of demonstrations, we have shown this method to be consistent with method (a). If the requirements of method (c) are violated significantly, diagnostics are printed in the main output file. Method (d) is an option we consider for the future, but it is not employed at the present time.

Carrier Runs vs. Sideband (Subcarrier) Runs

As we have said, the steady-state and resonance conditions are separate from one another. "Steady-State" refers to the quality of the convergence. "Resonance" refers to the nature of the steady-state which has been found. Most importantly for users of this program, be aware that:

The simulation program ALWAYS brings all fields to a steady state consistent with the tolerances and maximum allowed iterations as specified in **ligo.dat**.

The program DOES NOT always bring the lengths to resonance. This is an option that is turned on or off in **ligo.dat**. This option is the primary distinction in the program between carrier runs, and sideband (or subcarrier runs). (There is another distinction, an internal one in the program; it is discussed later in the section, "Understanding the Main Output File". It can be ignored for now.)

In a carrier run:

- All lengths are updated during each relaxation step, in order to satisfy our definition (b) of resonance and (hopefully) maximize the FP arm cavity power.
- In the end, the cumulative adjustments are written out to an auxilliary file, called **resonant_values_bal.dat** (for the balanced Michelson cavity) or **res_val_unbal.dat** (for the unbalanced case).

In a sideband run:

- The lengths are never updated or changed.
- These (fixed) lengths are read in from the auxiliary file produced by the preceeding carrier run.
- Given these lengths, the user must optimize the *frequency* of the sideband field (by trial and error searching, perhaps) to maximize the recycling cavity power while keeping the FP arm cavity power close to a minimum.

In conclusion, the carrier and sidebands are converged in *separate runs*, but the interferometer lengths are identical in each case. If you want to perform a run with a sideband frequency, you must therefore:

1. Set up **ligo.dat** to run the carrier first.
2. Set up **ligo.dat** to run the sideband case, and make sure the sideband run reads in the same auxiliary lengths file written out by the preceeding carrier run.

To explain how you can choose between carrier and sideband runs, we take an excerpt from the sample **ligo.dat** file in Appendix A.
On the next line after:

"**Bring exact lengths to resonance (1) or read fixed lengths from data file (0)?**"
there must be an integer: a 1 or a 0. A "1" indicates that it is a carrier run, and a "0" indicates that it is a fixed-length (i.e. sideband) run.
The next line reads:

"**Data file to (write/read) the resonant lengths, waist & radcin (to/from) :**"
What follows this line, both for carrier & sideband runs, is the auxiliary file for the converged lengths. This name must be matched for any two runs which are meant to take place in identical interferometers.

(One last note: If you have *ever* executed a particular carrier run, then every number that will appear in the auxliary length file will also appear somewhere in the body of its main output file (e.g. **ligo_bal_carr.out**, or **ligo_unbal_carr.out**). You may therefore edit the auxiliary length file instead of re-running the carrier frequency; but this only works if *none* of the interferometer parameters have changed from carrier to sideband run.)

Balanced vs. Unbalanced Inteferometer

Interferometer configuration will not be discussed in serious detail in this document. Nevertheless, an important configuration question relative to LIGO is whether its on-line and off-line flat mirrors will be equidistant from the input recycling mirror, or whether there will be a macroscopic offset, or imbalance.

The capability of generating an imbalance in the michelson arms, while still (for the carrier) holding the antisymmetric beamsplitter port to a dark fringe, has been built into the LIGO simulation program.

The place in the data file to input this imbalance is after the line,

"**Unbalancing length to add to L2 and subtract from L3:**"

L2 is the length between the recycling mirror and the on-line input mirror, and L3 is the distance between the recycling mirror and off-line input mirror. This imbalance is added on to the "base lengths" asked for earlier in the data file; even for a run with unbalanced arms, we commonly set the base lengths to equal values.

Run & Decomposition Templates

Given these 2x2 run possibilities, we have 4 general types or runs, and each of these types is associated with a key phrase that appears in all related filenames, as follows:

- "bal_carr" means length-symmetric interferometer, carrier run,
- "unbal_carr" means unbalanced interferometer, carrier run,
- "bal_sb" means symmetric interferometer, sideband run,
- "unbal_sb" means unbalanced interferometer, sideband run.

For each of these four run categories, we have written "templates" which contain the fundamental distinctions between these runs. Here is what's available:

Simulation main data file templates (to be copied into **ligo.dat** before running):

- **ligo_bal_carr.dat**
- **ligo_unbal_carr.dat**
- **ligo_bal_sb.dat**
- **ligo_unbal_sb.dat**

Cray-based scripts for simulation batch jobs (using the qsub command):

- **ligo_run** (runs with whatever is in **ligo.dat**)
- **ligo_bal_carr_run** (copies **ligo_bal_carr.dat** to **ligo.dat**, then runs)
- **ligo_unbal_carr_run** (same for **ligo_unbal_carr.dat**)
- **ligo_bal_sb_run** (same for **ligo_bal_sb.dat**)
- **ligo_unbal_sb_run** (same for **ligo_unbal_sb.dat**)
- **ligo_balanced_runs** (symmetric interfer., carrier then sideband)
- **ligo_unbal_runs** (unbalanced interfer., carrier then sideband)
- **ligo_carrier_runs** (symm. & unbal interfer., carrier only)
- **ligo_grand_run** (All 4 cases in order: carrier & sideband, symm. & unbal interfer.)

These scripts are submitted with the qsub command described earlier. You may have to allow for more than 1 hour in your job (on the "infinite-time" queue) to execute several runs as one job.

Be aware that if you use any of these batch scripts other than **ligo_run** , editing **ligo.dat** before you submit the job is useless; instead, you have to edit the relevant main data file templates appropriately. If you must do this, be sure to keep in existence a copy of the primitive version of each data file template, so that you do not lose track of all of your changes.

Mode decomposition script files for postprocessing of Cray data:

- **decomp_bal_carr** (carrier, symm. interf.)
- **decomp_unbal_carr** (carrier, unbal interfer.)
- **decomp_bal_sb** (s.b., symm interfer.)
- **decomp_unbal_sb** (s.b., unbal interfer.)
- **decomp_all** (All 4 cases together: carrier & s.b., symm. & unbal interfer.)

Before using any of the mode decomposition scripts, you must take the following steps to make sure you have all the necessary files:

1. Copy everything from
   ~brett/fft_documentation/fft_user_files/mode_decomposition
   into your own decomposition directory. (Do this only if you haven't ever done it before.)

2. Run the LIGO simulation on the Cray with any of the 4 main data file templates you wish to use.
3. Bring all of the created field files (12 per case) to your Sun decomposition directory.
4. Run (see below) the mode decomposition scripts which correspond directly to the main data file templates that you ran the Cray simulation program with in step 2.

To run any of these mode decomposition scripts:

A. Copy the first decomposition script you need (for one of the 4 cases of runs) into the file "**decomp_active**".
B. Open **decomp_active** with the text editor of your choice.
C. Observe the placeholder phrase "CHANGE_THIS_PART" at many places in the document. Decide upon a single phrase that you feel is informative and characteristic of this particular simulation run (e.g. "tilted_mirrors"). Replace the phrase "CHANGE_THIS_PART" with your characteristic phrase everywhere in this script.
D. Exit the editor, and run the script by typing the command: **decomp_active**
E. Wait 10 minutes or so.
F. Observe that twelve new files have been created, and are called, for example:
"**eins_tilted_mirrors.symm_carr_results**",
"**eref_tilted_mirrors.symm_carr_numbers**",
and so on. These ***results** and ***numbers** files contain all the information from the decomposition run.
G. Also observe that your twelve electric field files have been RENAMED, using your chosen phrase. The pattern is:
eins_bal_carr.re — —> **eins_tilted_mirrors_bal_carr.re** ,
eref_bal_carr.im — —> **eref_tilted_mirrors_bal_carr.im** ,
and so on.
H. It is *your responsibility* to rename your simulation main output file (perhaps originally called "**ligo_bal_carr.out**") to match the names of these decomposition results. It is recommended that they be stored together, perhaps by creating a separate subdirectory for the results of each new run.
I. Use the command "**enscript –2 –r –h –G <*filenames*>**" to print out the ***results** or ***numbers** files that you wish to look at.

J. Repeat from step A, with any other decomposition scripts that you need. Continue this procedure until all of your electric field files have been decomposed.
K. With all the "results" and/or "numbers" data in hand, you are now ready to study the physics of your simulated LIGO interferometer.

Physically Interesting runs with Mirror Imperfections and Tilts

Within the simulation, the mirrors may possess nontrivial physical properties. There are five such adjustable mirrors in the interferometer: the recycling mirror, the on-line input mirror, the off-line input mirror, and the on- and off-line back mirrors of the Fabry-Perot cavities. (The *beamsplitter* is not physically simulated; it is merely represented by a number which multiplies each pixel of a field as it enters one of the beamsplitter's ports.)

Important properties of these realistic mirrors are:

1. They have "baffles" which absorb any electric field that extends beyond the specified mirror diaphragms.
2. The recycling mirror & FP back mirrors are curved.
3. The recycling mirror & FP back mirrors may be displaced transverse to the beam.
4. All 5 mirrors may be rotated through 2 nontrivial angles.
5. All 5 may be given surface deformations of various kinds (Zernikes, etc.).
6. The recycling mirror & both input mirrors may be given realistically deformed substrates.
7. The curvatures, tilts, deformations, etc., are all compatible with each other in the approximation of (appropriately defined) *small perturbations*.

Comments:

- The recycling mirror radius of curvature is not given in **ligo.dat**. It is calculated in the simulation to achieve a proper matching condition.
- The back mirror radii of curvature *are* given in the **ligo.dat**, and they must be equal for complete beam matching to be possible.
- Any imbalance in the Michelson arms is ignored by the matching algorithm.
- Any tilt or deformation is handled by adding a small, position-dependent Δz to each pixel.
- In the small-distance approximation, each Δz is equivalent to a simple phase shift for that pixel.

- Any geometrical “interactions” between curvature, tilt, and deformation are ignored.

Important note: mirror deformations are not “anchored” to the mirrors; they are anchored to the 128x128 computational grids. In other words, the deformation in pixel (37,59) will remain at pixel (37,59) regardless of any tilting, transverse displacement or rotation which may cause that pixel to fall on a physically different point on the mirror. The baffles are *also* anchored the computational grid, and not to the center point of the mirror.

We use two methods for producing files with useful surface & substrate deviations:

a. Adaptation of real-world mirror deviation information, obtained from measurments, into properly formatted mirror descriptor files.
b. Utilization of a program capable of making surfaces with Zernike-polynomial deformations.

Using Real Mirrors for Simulated Surfaces and Substrates

The conversion of real-world mirrors into simulation-compatible mirror files must naturally be handled by the user of the program. The arrangement of pixels in each mirror file is as follows:

Each mirror file has 128 x 128 = 16,384 lines, one pixel deformation per line. The file is arranged, pixel-wise, as follows:

L=1,M=1
L=1,M=2
.
.
L=1,M=128
L=2,M=1
L=2,M=2
.
.
L=128,M=128

The pixel location in the plane of the beam is such that (x,y) is proportional to (L,M). The laser beam is normally centered at (L,M)=(64.5,64.5) (that is, in

between the 4 most centralized pixels.) The width of each pixel is designed to make the grid fill up the "window of calculation" as specified in **ligo.dat**.

An unavoidable discrepancy to note is that while in most simulation-supporting files the pixels are numbered 1–>128, PV-WAVE and all of the PV-WAVE plotting procedures number the pixels 0–>127.

Zernike Polynomial Mirror Deformations

The Zernike-making program, written by Partha Saha (in the C computer language), creates a Zernike of your choice with an adjustable amplitude and fixed physical radius of .125 m (same radius as the simulated LIGO mirrors), which is centered on the 128x128 grid. Copies of the C source file (**mk-zer.c**) and the executable code (**mk-zer**) are available on **kepler.mit.edu** in the subdirectory:
~brett/fft_documentation/fft_user_files/zernikes

The normalization and amplitudes of these Zernikes is given by this rule:
A scaled amplitude of (λ/N) for us means that the Zernike has an rms variation of (λ/N)/Sqrt(π).

(For a more complete explanation of the normalization of these Zernikes, you may preview the LaTex file **zernike_normalization.dvi**, which is available in this same subdirectory.)

Here is a sample utilization of the Zernike-making program. The computer's output is in plain text. What you must type is in **bold**.

> **mk-zer poly11 poly 31**
Input lambda (m): **5.14E-7**
Input amplitude scaling : **.00333333333333**
Input n1, l1 : **1 1**
Input n2, l2 : **3 1**
>

As a result, a Zernike polynomial with indices (1,1) is placed into the file **poly11**, and a Zernike (3,1) is placed into the file **poly31**. The Zernike's "scaled amplitudes" here are both (λ/300), where λ = 5.14 x 10^-7.

The Zernike (1,1) created above represents a tilt about the y-axis (orientation as defined in the fft program), and this will result in the creation of TEM_{10}, as reported by the mode decomposition program. To make a tilt about the x-axis

(thus creating TEM_{01}), substitute the indices **1 -1** for **1 1** in the above run of **mk-zer**. Furthermore, to change the *sign* of the tilt, you may use a negative "amplitude scaling" factor.

Angular Conventions For Mirror Tilts

Each mirror can be thought of as a "rigid body" with its center of mass fixed in place. To define the orientation of such an object, we require three angles of rotation. But because the mirrors (omitting deformations) are circularly symmetrical, one of these angles is meaningless, so we require only two angular parameters to specify the orientation of each mirror in the LIGO simulation.

The choice of angular parameters that can be used is not unique. We have found it convenient to use these parameters: theta, which is the angle (in microradians) between the laser beam propagation vector and the normal vector of the mirror surface, and phi, which is a *subsequent* rotation (in degrees) of the theta-tilted mirror about the axis parallel to the laser beam propagation vector (the "z-axis"). Here are some tilt examples to help you picture the angular conventions:

theta = 30, phi = 0 : A tilt of 30 microradians about the y-axis. (TEM_{10} will be created in the interferometer.)
theta = 30, phi = 90 : A tilt of 30 microradians about the x-axis. (TEM_{01} will be created.)
theta = 30, phi = 45 : A tilt of 30 microrad about the axis $\frac{1}{\sqrt{2}}(x+y)$ (Equal amounts of TEM_{10} & TEM_{01} will be created.)
theta = 0, phi = anything : Equivalent to no tilt at all, because of mirror circular symmetry.

Beam-Weighted Tilt Removal from Mirror Surfaces

A current limitation of the LIGO simulation program is that it does not optimize mirror tilts to bring the interferometer to resonance.

In general, a local mirror tilt that exists near the center of the beam will convert TEM_{00} into unwanted TEM_{10} and TEM_{01}. This will reduce the overall cavity gains, and these unwanted higher modes will corrupt the reflected and asymmetric-port electric fields.

Although programs exist to remove the overall tilt of these mirrors, that is not sufficient to remove the local tilt near the beam's center. One of us (B.B.) has

therefore written an auxilliary program which removes the local tilt (and the local piston displacement) in the beam's center, by rotating the mirror appropriately. The unwanted tilt (and piston) components are first determined by reflecting a pure TEM_{00} beam off of the mirror, and analyzing the TEM_{00}, TEM_{10}, and TEM_{01} produced in the reflected field.

The tilt-removal program is available on **kepler.mit.edu** in the subdirectory: **~brett/fft_documentation/fft_user_files/mirror_characterization**.
The executable code is named **bounce.x**, and the required data file is named **mirror_bounce.dat**. These files and runs are very similar to those of the modal decomposition process performed on the interferometer's output electric fields.

"Before" and "after" versions of a Hughes-Danbury mirror surface are also available there, and they are called, respectively, **real_mirror_ligo.dat** and **real_mirror_untilted.dat**. The modal characterization of the "before" mirror is always produce by the run, and for this example it has been placed into the files named **mirror_bounce.results** and **mirror_bounce.numbers** .

To provide a brief discussion of the mathematics and the results of the tilt-removal process, a publisher memo has been written by Yaron Hefetz, and the publisher main filename is:
~brett/fft_documentation/fft_user_files/mirror_characterization/mirror_realign_doc

Be aware that repeated tilt-removal runs make the mirror alignment better and better; specifically, we find that three tilt-removals are more than sufficient for badly deformed and tilted mirrors.

Section 4 Understanding The Main Output File

This section is concerned with interpreting the main output file of the LIGO simulation. The sample output file we will be discussing is called **ligo_bal_carr.out**, and a copy of it is located in Appendix B. It is the output obtained by running the simulation program with the **ligo.dat** input file located in Appendix A.

Upon examining this copy of the main output file, one can quickly verify that the first portion of it is indeed an echo of the data contained in the main input file. A routine check of this echoed information after each run is the first and best way to verify that you have no mistaken ideas about the job which you have run, which could lead you to a drastic misinterpretation of the results.

Here are some items in the output file you may wish to examine (The most significant items to monitor on every run are preceeded by a *****) :

Waist of main input beam : The beam waist size is calculated, given the lengths of the Fabry-Perot arm cavites and the radii of curvature of the FP cavity back mirrors. It is chosen so that the beam waist is placed at the flat input mirrors, and that the beam radius of curvature at the far end of each FP cavity matches the curvature radius of the back mirrors. Obviously, if the parameters of the 2 FP's differ, the beam cannot match properly into the full LIGO. The typical waist size (for carrier & sidebands) calculated from our LIGO parameters is 2.1511917313544E-2 m.

Matched input radius of curvature : Once the beam waist, all (macroscopic) cavity distances, and the radii of curvature of the FP back mirrors are all set, there is one free parameter left to match the beam into the Michelson part of the interferometer (i.e. the "recycling cavity") : this parameter is the recycling mirror radius of curvature. This value is calculated in the simulation, and written out here, in the main output file. The typical value is 666678.6666666 m. Note that in "unbalanced" interferometer runs (with an imbalance typically of .6 m), the imbalance is ignored in calculating this radius of curvature.

Len2corr1, Len3corr1, Len4corr1, Len5corr1 : To bring the interferometer fully into resonance, the distances between the mirrors must be altered during the

running of the program by fractions of a wavelength. Look again at Fig.1, the diagram of the full-LIGO interferometer, on page 4. There are 5 possible lengths to change: L_1–L_5, but only 4 of those lengths have physical significance. This is because any change in L_1 (distance bet. recycling mirror & beamsplitter) is physically indistinguishable from a common-mode change in L_2 & L_3 (distances between beam splitter & flat mirrors), since the beam-splitter is only being modelled as a multiplicative number. In the program, therefore, we have chosen the following conventions:

- The beamsplitter is "nailed down" in place.
- The distance between the recycling mirror & beamsplitter (L_1) is treated as fixed.
- Distances L_2–L_4 are changed to achieve resonance.

These four "Len#corr1" values, in fact, represent the preliminary changes in length for the interferometer, before the fields are relaxed. These numbers are written into the auxiliary data file (here called **resonant_values_bal.dat**), to be used for a future run with the same inter-mirror distances (e.g. for a sideband run.)

*

In first_guess, carrier_run= 1 : For a "carrier run", the interferometer is always either brought to resonance by changing the distances between the mirrors, or, alternatively, the lengths have somehow been set up so that the interferometer is already in a resonant configuration without length adjustments. (See the next item about "pre-relaxed power".)

Pre-Relaxed power inside, by recycling mirror: 36.2345532719
Pre-Relaxed power in on-line arm cavity : 2355.720420108
Pre-Relaxed power in off-line arm cavity : 2355.720420108

Before relaxing the fields fully, a semi-analytical first estimate is made of the power in certain places within the interferometer: just inside of the Michelson cavity by the recycling mirror, just inside the two FP cavities by the input mirrors, and at the asymmetric port of the beamsplitter. Note that these are 4 of the 6 places where the fully relaxed fields will be written out.

These power estimates are usually very close to the final values for carrier runs; but they are only qualitatively good for sideband runs. The variable above called "**carrier_run**" will equal 1 or 0, but the chosen value is determined by the program in this way: if you have chosen (in **ligo.dat**) that the cavity lengths are to be brought to resonance, then **carrier_run** always equals 1. If the cavity distances are fixed, and not brought to resonance, then carrier_run *will still equal 1* if these pre-relaxation estimates for the FP cavities are sufficiently high, indicating that perhaps the user has rigged the lengths to make the laser frequency automatically resonant in the interferometer.

The true significance of the value of this variable **carrier_run** is that it indicates whether or not the entire interferometer (including the FP arm cavities) will be resonant given all of the specified input options. Depending on the value of **carrier_run**, the program chooses between two convergence algorithms, which are appropriate for the two different resonant conditions. (We will not, however, go into further detail about convergence methods at this time.)

*

Relaxed by iteration = 69: This line appears if all the fields in the simulation have reached a steady-state solution (within the allowed tolerances) in no more than the maximum # of iterations allowed in **ligo.dat**. If the target tolerances have not been reached by the calculation sigmas, but the limit on iterations *has* been reached, then the program will stop and the output file will instead contain the line: "**Completed max. # of iterations = <*the appropriate #*>**" This does not mean that the output results are no good. That conclusion depends upon the final sigmas of the relaxation. (See the next item.)

*

Fields in recyc. cav, arm cavities 4,5 relaxed to:
9.9866723750907E-7, 2*1.7111378365479E-6: These are the sigmas which represent the deviation of the final output fields from a steady-state solution. Even if the simulation has stopped because it ran out of iterations, as discussed above, *it is still likely that the run was good*, if the sigmas listed here are low enough. There is nothing magical about the convergence tolerances we have chosen; they are, in fact, usually not attained for our runs. As a general rule, any sigma greater than 10^-3 is not really good. A good rule of thumb is to remember that a sigma of 10^-N indicates that the amplitude of the corresponding output field is only

good to about 10^-N. (However, this is a hand-waving rule; it is always better to converge to smaller sigmas, and to be conservative in your estimates of how many of the significant figures in your results are accurate.)

Len2corr1, Len3corr1, Len4corr1, Len5corr1 : These are the final adjustments to the cavity distances, which take into account the full relaxation of the cavity fields. Like the "len#corr1" numbers, these are recorded in the auxiliary data file (e.g. **resonant_values_bal.dat**) for possible use in future runs.

*****

Input power : 1.
Reflected power : 4.3355567438685E-2
Power inside, by recycling mirror : 36.23451739523
Power by . . .

The powers of the various simulated fields are always of interest. These power values correspond to the relaxed electric fields written out by the program, which may be analyzed via mode decomposition on the Sun computers. (Only the input laser beam is never written out, because it is always a perfect TEM_{00} with unit power.)

*****

1-C = 2*powasymm/(powsymm+powasymm) = . . . : this is our definition of the contrast defect of the interferometer. For carrier runs (i.e. fully-resonant interferometer runs), this quantity has great physical significance.

Grand total cpu time used (sec) : 100.5420300162 : At a rate of about 50$ per Cpu hour, the total cost of this particular run is about 17¢. In general, the longest runs we have performed have lasted about 30 Cpu minutes.

Diagnostics and Warnings Which May Appear in the Main Output File

The round-trip cavity phase shifts are FARTHER FROM ZERO THAN EXPECTED:
Round trip phase, on-line arm cavity: 1.1684820025324E-10
Round trip phase, off-line arm cavity: –9.2446127396113E-11
Round trip phase, entire on-line path: 2.1361244417396E-10
Round trip phase, entire off-line path: 2.0043955811202E-10

As discussed previously in this manual, a steady-state, resonant field that undergoes a cavity round-trip must return to its starting point (nearly) spatially identical and *in phase with* the pre-round-trip field. The "Round trip phase" numbers above are combined measures of how much the cavity fields are not really fully relaxed, and how far off resonance the inter-mirror distances are from a completely resonant configuration. Such phases are given here for several different round-trip routes through the interferometer.

These phases are "normalized" to a wavelength, in the sense that a completely off-resonant cavity would show up here with a phase shift value of exactly π. These sample numbers are very small, but still large enough to trigger the printed diagnostic. You must use your judgement on whether the resulting interferometer is sufficiently on resonance to provide meaningful output numbers.

Phase shifts upon reflection from the resonant cavities deviate FARTHER FROM PI THAN EXPECTED:
At recycling mirror, phase bet. resonant field reflection and prompt reflection is: pi + –3.3294947237271E-4
At on-line input mirror, phase bet. resonant field reflection and prompt reflection is: pi + –1.7137282639238E-8
At off-line input mirror, phase bet. resonant field reflection and prompt reflection is: pi + 2.1843746736298E-7

These phases are similarly based on a principle of resonance, namely, that a field reflected from a resonant cavity should experience an extra π of phase shift, compared to the field which has immediately reflected off of the input mirror of the cavity (the "prompt" reflection). Although we have not proven that this definition is completely equivalent with our round-trip definition of resonance for

the most general interferometer with perturbed mirrors, we take it as a good sign when the resonant reflections from both FP input mirrors (for the carrier) and from the recycling mirror are shown to have experienced a phase shift of very close to π.

The following phase is FARTHER FROM PI THAN EXPECTED:
At recycling mirror, phase bet. resonant field reflection and prompt reflection is: pi + 1.0024823736762E-3

This is the same diagnostic as the previous one, except that it is for a sideband run, where only the Michelson-part of the interferometer is expected to be resonant.

WARNING: Sum of Matrix Inversion Errors = 6.1054416848205E-10 in iter 83
WARNING: Sum of Matrix Inversion Errors = 6.5202543539554E-10 in iter 84
WARNING: Sum of Matrix Inversion Errors = 6.8113244612449E-10 in iter 88
WARNING: Sum of Matrix Inversion Errors = 5.7396030075049E-10 in iter 89

These error messages refer to problems the fft program is having in the process of finding the steady-state fields. If these messages appear, but your run converges anyway, then ignore the warnings. If, however, the number quoted in these warning increases without bound, then your run will never converge, and you must run with a less severely perturbed interferometer (i.e. smaller mirror tilts), if you wish to achieve convergence.

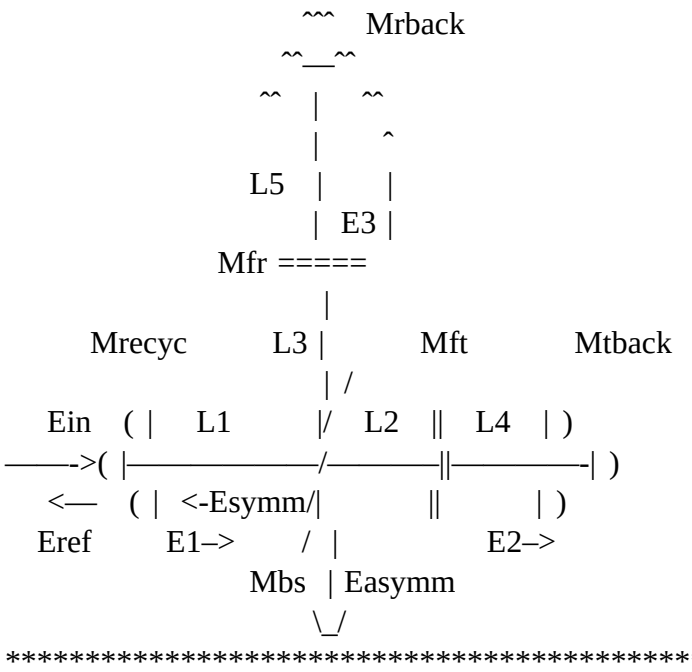
No convergence in 30 SVDCMP inversion iterations, iter = <*Number*>

As long as your run has terminated well, with good sigmas, ignore this message completely.

We have finally come to the end of the chapter on the mechanics of running the fft simulation program. Good luck with your runs, and if you experience unsolvable difficulties of any kind, feel free to contact either of us, Brett Bochner and Yaron Hefetz, for assisstance.

Appendix A Main Input Data File for a "Perfect" LIGO Interferometer (ligo.dat)

Input data for simulation of full LIGO interferometer
==========



****
Name of main output file: (max 25 characters)
ligo_bal_carr.out

Wavelength of the laser (all lengths are in meters):
5.14D-07

Frequency added to laser frequency for modulation (Hz):
0.0

Base Lengths L1, L2, L3, L4, L5:
8.5 3.5 3.5 4000.0 4000.0

Unbalancing length to add to L2 and subtract from L3:
0.0

Bring exact lengths to resonance (1) or read fixed lengths from data file (0)?
1

Data file to (write/read) the resonant lengths, waist & radcin (to/from) :
resonant_values_bal.dat

Radii of curvature of back mirrors Mtback, Mrback:
6000.0 6000.0

Width of square calculational window:
0.35

Radius of diaphragms on Mrecyc, Mft, Mfr, Mtback, Mrback:
.125 .125 .125 .125 .125

Intensity Reflectivities of Mrecyc, Mft, Mfr, Mtback, Mrback:
0.9599 0.97 0.97 0.9999 0.9999

Loss in mirrors Mrecyc, Mft, Mfr, Mbs:
1.0D-4 1.0D-4 1.0D-4 1.0D-4

Position & angle error of excitation laser beam: dx,dy,theta(microrad),phi(deg)
0.0 0.0 0.0 0.0

Position & angle error of mirror Mrecyc; dx, dy, theta(microrad), phi(deg)
0.0 0.0 0.0 0.0

Angle error of mirror Mft; theta (microradians), phi (degrees)
0.0 0.0

Angle error of mirror Mfr; theta (microradians), phi (degrees)
0.0 0.0

Position & angle error of mirror Mtback; dx, dy, theta(microrad), phi(deg)
0.0 0.0 0.0 0.0

Position & angle error of mirror Mrback; dx, dy, theta(microrad), phi(deg)
0.0 0.0 0.0 0.0

Base phase thicknesses of mirrors Mrecyc, Mft, Mft :
0.0 0.0 0.0

Maximum # of relaxation iter's, and max. integ. errors (recycling cav, cav4,5):
1200 1.0D-6 1.0D-5 1.0D-5

Names of the files containing variations in the mirrors ("+" means blank):
Substrate of Mrecyc : (max 25 characters for all mirror filenames)
+
Reflective side of recycling mirror :
recyc_perf.dat
Substrate of on-line input mirror :
+
Reflective side of on-line input mirror :
online_inp_perf.dat
Substrate of off-line input mirror :
+
Reflective side of off-line input mirror :
offline_inp_perf.dat
Reflection from on-line FP back mirror :
online_back_perf.dat
Reflection from off-line FP back mirror :
offline_back_perf.dat
File names of output fields. First real part, then imaginary. (max 25 chars)
For field Ein (see diag. at top):
phiin_bal_carr.re

phiin_bal_carr.im

For field E1:
eins_bal_carr.re

eins_bal_carr.im

For field E2:
eflatt_bal_carr.re

eflatt_bal_carr.im

For field E3:
eflatr_bal_carr.re

eflatr_bal_carr.im

For field Esymm:
esymm_bal_carr.re

esymm_bal_carr.im

For field Easymm:
easymm_bal_carr.re

easymm_bal_carr.im

For field Eref:
eref_bal_carr.re

eref_bal_carr.im

Appendix B Main Output File for a “Perfect” LIGO Interferometer (e.g. ligo_bal_carr.out)

```
====================================
        Output file for LIGO Interferometer
====================================
Laser Carrier Wavelength (m) :  5.14E-7
Frequency added to laser frequency for modulation (Hz):  0.
Wavelength of sideband (if necessary) :  5.14E-7
Base cavity lengths :  8.5, 2*3.5, 2*4000.
Unbalancing length added to L2, subtracted from L3 :  0.
Bring exact lengths to resonance (1) or read fixed lengths from data file (0)?
1
File for stored lengths, waist & radcin:  resonant_values_bal.dat
Modified L2, L3:  2*3.5
Radii of curvature R4,R5 :  2*6000.
Width of square calculational window:  0.35
mirror diaphragms :  5*0.125
Mirror intensity reflectivities :  0.9599, 2*0.97, 2*0.9999
Losses in recycling mirror, on-line & off-line input mirrors, and beam splitter:
4*1.E-4
x,y,theta(microrad),phi(deg) offsets of excitation laser beam:  4*0.
dx,dy,theta(microrad),phi(deg), input mir :  4*0.
theta(microrad), phi(deg), for mirror Mft :  2*0.
theta(microrad), phi(deg), for mirror Mfr :  2*0.
dx, dy, theta(microrad), phi(deg), for mirror Mtback :  4*0.
dx, dy, theta(microrad), phi(deg), for mirror Mrback :  4*0.
Phase thicknesses of Min,Mft,Mfr :  3*0.
maxm no. of iters. & max integ. errors (cavs 1,2,3) :
1200, 1.E-6, 2*1.E-5

Mirror files used :
Substrate of M-recyc :  +
ref. side of M-recyc :  recyc_perf.dat
```

Substrate of Mft : +
ref. side of Mft : online_inp_perf.dat
Substrate of Mfr : +
ref. side of Mfr : offline_inp_perf.dat
Reflection from Mtback : online_back_perf.dat
Reflection from Mrback : offline_back_perf.dat

Output files for real & imag parts of fields:
phiin_bal_carr.re
phiin_bal_carr.im
eins_bal_carr.re
eins_bal_carr.im
eflatt_bal_carr.re
eflatt_bal_carr.im
eflatr_bal_carr.re
eflatr_bal_carr.im
esymm_bal_carr.re
esymm_bal_carr.im
easymm_bal_carr.re
easymm_bal_carr.im
eref_bal_carr.re
eref_bal_carr.im
———————————————————

****
Begin Relaxation (bringing lengths to resonance)
****
Waist of matched input beam : 2.1511917313544E-2
Matched input mirror radius of curvature : 666678.6666666
len1 – len5 :
5.0400012696628E-7, 2*2.6800003638527E-7, 2*1.6203557606786E-7
Len2corr1, Len3corr1, Len4corr1, Len5corr1 :
2*-6.5309349159129E-10, 2*1.731147119962E-7
In first_guess, carrier_run= 1
Pre-Relaxed power inside, by recycling mirror: 36.2345532719
Pre-Relaxed power in on-line arm cavity : 2355.720420108

Pre-Relaxed power in off-line arm cavity : 2355.720420108
Pre-Relaxed power from asymmetric port : 0.
Relaxed by iteration = 69
Fields in recyc. cav, on-line and off-line arm cavities relaxed to:
9.9866723750907E-7, 2*1.7111378365479E-6
Len2corrs, Len3corrs, Len4corrs, Len5corrs :
2*-2.0505075489345E-19, 2*4.8487050807775E-18

Input power : 1.
Reflected power : 4.3355567438685E-2
Power inside, by recycling mirror : 36.23451739523
Power by flat in on-line arm resonance cavity : 2355.71844169
Power by flat in off-line arm resonance cavity : 2355.71844169
Power from symmetric beamsplitter port: 35.28151279965
Power from antisymmetric beamsplitter port : 0.
1-C = 2*powasymm/(powsymm+powasymm) = 0.
alpha1 = 1 - (powins/36.235) = 1.3318746326263E-5
alpha2 = 1 - (powonline/2355.72) = 6.6150054323089E-7
alpha3 = 1 - (powoffline/2355.72) = 6.6150054323089E-7
Grec = pow_E1/pow_laser = 36.23451739523
Gcav5 = pow_E5/pow_laser = 2355.71844169
Gcav9 = pow_E9/pow_laser = 2355.71844169
==========
Cpu time in initphase (sec) : 0.878279114592
Total cpu time in relax (sec) : 73.56686037221
Cpu time spent in findabc (sec) : 4.5646546284
==========
==========
Grand total cpu time used (sec) : 89.61113017886
==========

Appendix C Runs Performed With The Full-LIGO Simulation Program

For any choice of mirrors, there are four cases of runs that may be performed :

A. Carrier frequency, Balanced Michelson cavity (input mirrors equidistant from recycling mirror).
B. Carrier frequency, .6 meter Imbalance in Michelson cavity
C. Sideband frequency, Balanced Michelson cavity
D. Sideband frequency, .6 meter Imbalance in Michelson cavity

Runs Performed:

- All Mirrors Perfect (cases A-D). Results agree with semi-analytical power calculations.
- λ/300 Zernike Polynomials[1] on various mirrors; one Zernike-perturbed mirror per run. Zernike types (8,0),(9,3),(14,0),(11,9),(10,0),(7,1). (run cases A-D)
- Double Zernike perturbations: λ/300 Z(12,0) on on-line flat, λ/300 Z(7,1) on off-line flat. (cases A-D)
- Hughes-Danbury (HD) "real mirror"[2] on the on-line flat mirror. (cases A-D)
- Perkin-Elmer (PE) "real-substrate"[3] simulated inside the off-line flat mirror. (cases A-D)
- Runs with "Fake mirrors #0–4"[4].

  a. λ/300 RMS Fake Mirror runs. All combinations, from a single fake mirror (w/others perfect) to all five fake mirrors in use simultaneously. (cases A-D)
  b. λ/900 RMS Fake mirror run; 5 fakes in use simultaneously. (case B only)
  c. λ/900 RMS Fake mirrors with beam-weighted tilt removed from them. 5 fakes in use simultaneously. (cases B,D)

---

1 As noted in the section on Zernike polynomials in Chapter 2, a (λ/N) Zernike refers to a Zernike polynomial with an rms of (λ/300)/Sqrt(π)

2 We have adapted the surface phase delay map of an existing mirror (data received from by Hughes-Danbury) to our program. This real mirror surface may be placed on any simulated mirror.

3 We have adapted the phase delay map of the substrate of an existing mirror (data received from by Perkin–Elmer) to our program. This real mirror surface may be placed on any simulated mirror.

4 Yaron Hefetz has prepared a series of "fake mirrors" (#'s 0–4) with the same power spectrum as a smoothed version of the HD real mirror, but with randomized phases between components. Fake mirror #0 is the has not been phase-randomized; just smoothed.

d. All 5 λ/900 RMS fake mirrors in place; used random, small perturbations to cavity arm lengths to verify resonance-finding capability of simulation program.

- Test of "healing" by using a Zernike on one flat mirror, and changing the recycling mirror power transmission. These runs were performed:

  λ/300 Zernike (10,0), T_recyc=4% (normal value), 40%, 80%. (Cases A,C)

  We observed no changes in the contrast defect in these runs with different recycling mirror transmission coefficients.
- Longer Michelson Cavities (cases A,C).

  a. Michelson arm = 2503.5 m :

     - Hughes-Danbury (HD) "real mirror" on on-line flat mirror. Both flat mirror w/small aperture radii (.062 m).

  b. Michelson arm = 3003.5 m :

     i. All Perfect Mirrors
     ii. HD real mirror on on-line flat.
     iii. HD real mirror on on-line flat, and small flat mirror aperture radii (.062 m).
     iv. Fake Mirror #1 on on-line flat; small mirror aper's (.062 m).

- HD real mirror on on-line flat. Small aperture radii (.062 m) on both flats. (cases A-D)
- HD real mirror on on-line flat. Small recycling mirror aper. radius (.043 m). (cases A-D)
- Comparisons of FFT program with the Analytical LIGO Simulation Program (written by Yaron Hefetz & Nergis Mavalvala), also called the "Modal Decomposition Program". Runs performed (all are Case B) :

  1. No tilts, Perfect Interferometer.
  2. θ_x(on-line input mir.) $= 12.5\times10^{-9}$ radians.
  3. θ_y(on-line input mir.) $= 12.5\times10^{-9}$ radians.
  4. θ_x(on-line input mir.) $= -\theta_x$(off-line input mir.) $= 2.5\times10^{-9}$ radians.
  5. θ_x(on-line input mir.) $= -\theta_x$(off-line input mir.) $= 12.5\times10^{-9}$ radians.
  6. θ_x(recycling mir.) $= \theta_y$(recycling mir.) $= \frac{5}{\sqrt{2}}\times10^{-9}$ rad, θ_x(on-line back mir.) $= 10\times10^{-9}$ rad, θ_y(off-line input mir.) $= -7\times10^{-9}$ rad.

In all of these runs, the agreement between the analytical and FFT numerical results was excellent.