
End to End Simulation Program for Gravitational-Wave Detectors

H. Yamamoto, B. Bhawal, M. Evans, E. Maros, M. Rakhmanov,
R.L. Savage, Jr

*LIGO Laboratory, California Institute of Technology, Pasadena CA,
91125, USA, yamamoto_h@ligo.caltech.edu*

G. Cella

*Dipartimento di Fisica, Universita' di Pisa, piazza Torricelli 2,
56100- Pisa, Italy*

S. Klimenko

*Department of Physics, University of Florida, Gainesville, FL, 32611,
USA*

S. Mohanty

*Center for Gravitational Physics and Geometry, The Pennsylvania
State University, University Park, Pennsylvania, 16802, USA*

Abstract

Within the LIGO project, a time domain simulation program has been developed to provide an accurate description of gravitational wave detectors that use laser interferometry. The program simulates the time evolution of fields, optics, mechanical structures and control systems. The program is written in C++ and the modular design of this program makes it possible to simulate a wide variety of processes without modifying the program. This design also makes it easy to add new functionality. Using a graphical user interface (GUI), one can easily set up or modify the system to be simulated. The underlying physics and the structure of this simulation package are described in this paper.

1. Introduction

Experiments in large scale gravitational wave (GW) detection using laser interferometry are entering a new observational era. These include the LIGO project [7], the Laser Interferometer Gravitational-Wave Observatory. As has been demonstrated in the field of high energy physics, a detailed computer simulation is crucial in order to operate and analyze a large complex experiment

successfully. A time domain simulation program for LIGO, the End to End simulation software package (called e2e hereafter), has been developed at the LIGO Laboratory in collaboration with scientists from other institutions [5]. The VIRGO project has developed a similar simulation program called SIESTA [2].

The e2e software has been developed mainly for the following purposes: (1) detector design; (2) detector diagnostics during the commissioning and operation phase; (3) pseudo data production for data analysis.

The program has been designed and developed within the following guidelines: (1) It should be easy to use e2e without prior programming experience; (2) It should be easy to create, modify and maintain the simulation configuration setup even for complex systems, such as LIGO interferometer; (3) It should be easy to simulate a wide variety of hardware configurations by using built-in software components, without modifying the source code; (4) The software structure should be such that maintenance and extension can be done easily and in a streamlined way.

In this paper, a brief explanation of e2e is presented. First, the physics ingredients and the software organization of e2e are explained. Then, a simulation of a Fabry-Perot (FP) cavity is used as an example to explain the usage of e2e. Lastly, the current status and future plans for e2e are discussed.

2. Physics in the End to End simulation package

2.1. Digitized time evolution for time domain simulation

The time evolution is simulated by breaking time into short intervals with time steps of $10^{-7} - 10^{-3}$ seconds, depending on the processes involved in the simulation. At each time step, causal data flows are followed in order to generate the requested data. When there is a closed loop involved in the simulation, the chosen time step should be small enough so that the artificial delay caused by the discrete-time simulation does not introduce spurious observable effects.

Linear systems are simulated using digital filters, whenever possible. For example, motion of single suspended mirror can be simulated using a digital filter representing the pendulum transfer function with the suspension point motion and the force acting on the mirror as inputs. Most linear servo systems can be implemented using digital filters.

Linear processes for which the digital filter implementation is not appropriate and non-linear processes are handled individually. However, the modular design discussed below makes it relatively easy to implement these processes.

2.2. Time domain modal model for field and optics simulation

The evolution of the field and its interaction with optical media are calculated using a modal model [6]. The field is expanded using the Hermite-Gaussian eigenstate solutions of the paraxial wave equation. The eigenstates are characterized by the waist size, w_0 , and the distance to the waist, z . When the field goes through a lens, w_0 and z are changed, while when the field interacts with mirrors, the field change is calculated perturbatively using the same eigenstate.

The e2e software can be used to simulate any planar optics configuration by combining mirrors and field propagators without modifying the source code. This method is flexible, but it suffers from a speed penalty when the frequency of interest is much lower than the inverse of the characteristic time scale of the system (e.g., the characteristic time of the recycling Michelson cavity of LIGO is $\sim 1/(30\text{MHz})$, while the frequency range of the GW and control signals is up to a few times 10kHz). The simulation speed of this kind of system can be improved by using an approximation method[10]. In the approximation, the field evolution over multiple reflections is calculated analytically by assuming all inputs change linearly during each time step. In e2e, three kinds of cavities have been implemented using this technique: FP, triangular and recycled Michelson cavity.

2.3. Simulation of Mechanics systems

The simulation of mechanical systems involves three elements: the seismic ground motion, seismic isolation systems (SEI) and the suspension systems (SUS). The first implementation is based on the following components.

At each LIGO site, the seismic ground motion is measured at various locations and the power spectra of those data are used. Seismic motions are synthesized using digital filters based on these power spectra [4].

The fabricator of the LIGO-I SEI developed a MATLAB code that calculates the transfer functions among various degree of freedom. Some of them have also been measured for the as-built systems. These transfer functions have been parametrized, and SEI can then be simulated using digital filters [4].

LIGO-I uses a single pendulum for the SUS design. A few separate models have been built [3,8,9] and were compared to validate each other. A simple model is used for the baseline implementation of a 3-D suspended mirror [9]. The thermal noise is modeled in an ad hoc manner using a model by S.Finn [5].

In order to simulate more complex systems, a new fully 3-D simulation code has been developed, the Mechanics Simulation Engine (MSE) [3]. It uses a basic set of fundamental mechanical objects (masses, beams etc.) that are combined to represent complete mechanical systems. The model for any mechanical object can be refined, according to the need for detail to simulate specific effects, such

as the detailed high frequency behavior. The code also provides methods to automatically improve the internal representation of the system.

The mechanical simulation starts by finding the equilibrium point of the configuration, then linearizes the dynamics of the system around this point. The matrix equation which represents the system becomes larger when the mechanical system is complex, or described in detail. However, the basic rules to construct the matrix are well formulated and, once validated for simpler configurations, the validity of the matrix for more complex configurations can be guaranteed.

3. Software infrastructure

The e2e software uses C++ and is modular in design. It consists of two major components: the simulation engine and the GUI.

The simulation engine performs the time domain simulation as described earlier. The program can be viewed as a software toolbox, and compound systems can be simulated by combining building blocks called modules (e.g., *mirrors*, *propagator*, etc). The configuration to be simulated and its parameters are described by simple syntax. A wide variety of configurations can be simulated without modifying any source code.

With this modular design, each module can be defined and implemented independently of other modules. When the program starts and a configuration file is read, the necessary modules are loaded and placed in an ordered execution queue. This modular design also makes it easy to add new functionality.

The engine is based on a C++ class library Adlib (*Adlib, the Digital Instrument Builder*). The basic object in Adlib is a module. Examples are *laser generator* and *mirror*. Most of them map into specific hardware. One special module, *box*, is like a function in C. Multiple modules, including other *boxes*, may be nested together into one *box*, which can in turn be used as a module.

Each module is characterized by the following quantities:

- specific inputs given by name, data type and default values, and outputs by name and data type.
- time independent parameters that specify the object.
- a function that initializes the object for a given set of parameters, and an action function which calculates outputs given the inputs at a given time.

For example, the module *mirror* has two fields, mirror position and orientation as inputs and two outgoing fields as outputs, has several parameters such as reflectance and transmittance, and has an action function which calculates the outgoing fields based on the incoming fields and the mirror placement.

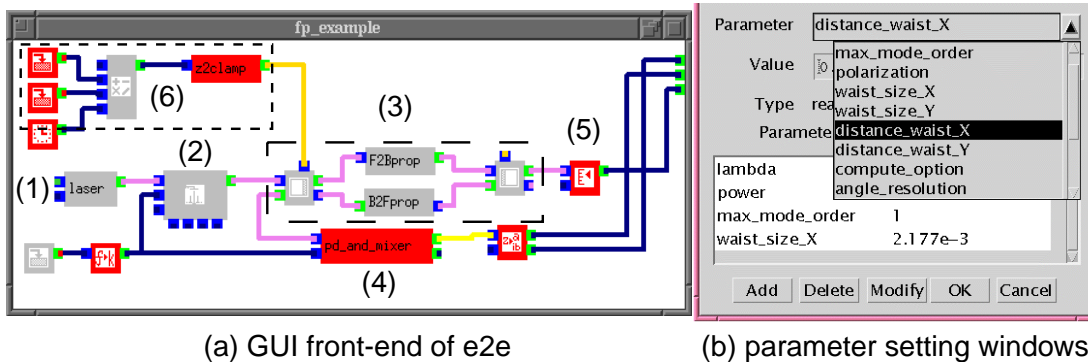


Fig. 1. Setup for Fabry-Perot cavity in e2e.

In order to create and modify the description files for the simulation engine, a program, *Alfi* (*AdLib Friendly Interface*), has been developed. With this program, the description file can be edited using a GUI. It is easy to start using the e2e program without learning the syntax of the description files and the inputs and outputs of various modules. *Alfi* also makes it possible to develop complex configurations which are otherwise difficult to handle.

4. Fabry-Perot simulation as an example

In this section, the usage and functionality of e2e is explained using a FP cavity as an example. Fig.1(a) shows the *Alfi* window for the example configuration. An iconic interface is used to insert modules into the window and to link inputs and outputs of modules to define data flow. For each module, parameters can be set using a dialog box such as that shown in Fig.1(b).

The process in Fig.1 is as follows. Module(1) generates a laser field, and it enters an Electro-Optic Modulator (EOM)(2). The number of sidebands can be specified, and by placing *EOM* modules in parallel or series, more complex phase or amplitude modulation schemes can be simulated. The field then enters the FP cavity(3). The two objects at each end of (3) are *mirror* modules. They are connected by *propagators*, left to right and right to left. The field reflected by the cavity goes to module(4) which acts as a photodiode and demodulator. The demodulation frequency can be specified, and photon shot noise can be turned on or off. The output of this module provides the demodulated signal. Module(5) is a power meter which gives the power transmitted through the cavity. The ports at the right edge of the window provide the output. They can be passed to other modules or can be saved to a file. Modules(6) models a linear motion, and the time dependent position is passed to one *mirror*. Any part of this window can be

created separately and saved, and can be included as a *box* module.

After the configuration is completed, it is saved as a text file. When the simulation program starts, it reads this file and generates the time series data.

5. Status

A 2-km FP cavity experiment began at the LIGO Hanford Observatory in November, 1999. The e2e software was used to study length control lock acquisition. Also, it was used to understand the complex fringe structure during the lock acquisition phase, and to understand various higher order harmonic signals caused by the non-linear response of the optics systems in the in-lock state.

In order to simulate the full LIGO interferometer, one must build a setup for the specific hardware configuration. Efforts toward this goal are reported elsewhere in this proceedings [11]. Work is also going on to model effects such as thermal lensing which is important for LOG experiment.

As high energy physics simulations have shown, the international consensus and collaboration in the development of simulation programs are necessary. LIGO Laboratory encourages efforts toward this end.

This work was supported by the National Science Foundation under Cooperative Agreement PHY-9210038.

6. References

1. Bhawal B. et al. 1997, LIGO document T970193, and references there in.
2. Caron B. et al. 1995, Nucl. Instrum. Methods Phys. Res., A 360, 375
3. Cella G. 1999, LIGO document T990106, T990107, T990108.
4. Daw E. 1999, LIGO document T990112.
5. Finn L.S., private communication, see also <http://gravity.phys.psu.edu/~lsf/SimData/>.
6. Hefetz Y. et al. 1997, J. Opt. Soc. Am. B 14, 1597.
7. Lazzarini A. et al. 1999, TAMA Workshop proceedings.
8. Mohanty S. 1999, LIGO document T990014.
9. Rakhmanov M. 2000, Ph. D thesis, California Institute of Technology.
10. Redding D. 1996, LIGO document T960171. Bhawal B. 1998, J. Opt. Soc. Am. A 15, 120.
11. Savage, Jr R.L. et al. 1999, TAMA Workshop proceedings.