CNRS
Centre National de la Recherche Scientifique

INFN
Istituto Nazionale di Fisica Nucleare

SIESTA, a Time Domain, General Purpose Simulation Program for the VIRGO Experiment

B.Caron
LAMII/CESALP, Annecy, France

L.Derome, R.Flaminio, X.Grave, F.Marion, B.Mours, D.Verkindt
Laboratoire de Physique des Particules (LAPP), Annecy-Le-Vieux, France

F.Cavalier
Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud, Orsay, France

A.Viceré
Dipartimento di Fisica dell'Università e INFN Sezione di Pisa, Italy

submitted to *Astroparticle Physics*

VIR-TRE-LAP-5700-101
Issue: 1
Date : August 25, 1998

VIRGO * A joint CNRS-INFN Project
Project Office: INFN-Sezione di Pisa * Via Livornese, 1291-56010 San Piero a Grado, Pisa Italia.
Secretariat: Telephone (39) 50 880 327 or 880 352 * FAX (39) 50 880 350 * e-mail virgo@pisa.infn.it

SIESTA, a Time Domain, General Purpose Simulation Program for the VIRGO Experiment

B.Caron

LAMII/CESALP, Annecy, France

L.Derome, R.Flaminio, X.Grave, F.Marion*, B.Mours, D.Verkindt

Laboratoire de Physique des Particules (LAPP), Annecy-Le-Vieux, France

F.Cavalier

Laboratoire de l'Accélérateur Linéaire, Université Paris-Sud, Orsay, France

A.Viceré

Dipartimento di Fisica dell'Università e INFN Sezione di Pisa, Italy

Abstract

In this paper we present the simulation program that has been developed for the VIRGO gravitational wave detection experiment. Although this program – SIESTA – is still evolving, it has reached a stage where the design requirements have been largely fulfilled. We first remind the needs and the choices which have steered the program design and led to the present structure. The contents of the program is then reviewed, the performances are discussed, and some typical applications are briefly described.

submitted to *Astroparticle Physics*

*Corresponding Author

1 Introduction

The construction of the VIRGO gravitational wave detector has been under way for a few years, and is led by a collaboration supported by both CNRS (France) and INFN (Italy). To probe gravitational waves, the detector uses suspended mirrors forming a 3 km long Michelson interferometer. The layout of the detector has been described in details elsewhere [1]. The apparatus is complex in many respects. Its sensitivity is determined by its design as far as optics, mechanics and electronics are concerned.

The mechanical system used to suspend and isolate from ground each test mass is a multiple pendulum with seven stages, including a precise positioning device. The Michelson interferometer used to extract the signal contains Fabry-Perot cavities in the arms, and uses the technique of light recycling. The optical configuration is further complicated by the use of a high frequency phase modulation and demodulation technique of the laser light. The optical response of the interferometer is analyzed on-line to react on the test masses through servo-loops, in order to maintain the interferometer at a working point.

These few considerations illustrate the need for a global, integrated simulation program of the equipment to assist the design of the control system as well as the learning process when the detector is being commissioned. Similar development efforts have been undertaken in related experiments [2].

Another major goal was to build a tool for producing simulated data, in order to develop, implement and run data analysis techniques. This implies the capability to describe both the detector behavior and the signals emitted by gravitational wave sources.

In section 2 we describe the general structure of the program. We then start in section 3 by reviewing the general use modules, namely those dealing with signal processing, output graphs, and the interface to the data formatting software. Sections 4 and 5 are devoted to the main aspects of the detector simulation: its mechanical behavior and its optical response. The simulation of gravitational wave signals is the subject of section 6. Finally a few representative examples of the program applications are shown in section 7.

2 General Structure

2.1 Design Issues

One of the design choices of the SIESTA[1] software has been to develop the simulation in the time domain. This is important for data analysis since it allows to produce the same type of simulated data as real data. It makes it also easier

[1]Simulation of Interferometric Experiments Sensitive To grAvitational waves

to simulate feedback loops, and to take into account any non-linear effect, for instance in the interferometer optical response or in the electronics. The choice of working in the time domain makes it also possible to reuse part of the code in the online system.

SIESTA is written in the C language, and is based on an object oriented structure which is most suited to build an integrated simulation involving many different aspects. Although an object oriented language such as C++ might seem more appropriate, standard C was preferred when the software development started seven years ago, for reasons of performances, standardization and portability. The latter issues are important ones in a wide collaboration for a program which has to be run on many different kinds of platform.

The implemented framework is flexible enough so that for a given issue only the relevant aspects of the detector have to be explicitly simulated, and the appropriate detail level (whenever several are available) can be selected. For instance, for most of data analysis purposes, it is sufficient to integrate in the simulation the generation of gravitational wave events and an effective noise generator reproducing the VIRGO sensitivity curve. On the other hand, simulating the control system requires to simulate both the mechanical and optical response of the system in detail. Figure 1 shows a summary of the different topics covered by SIESTA.

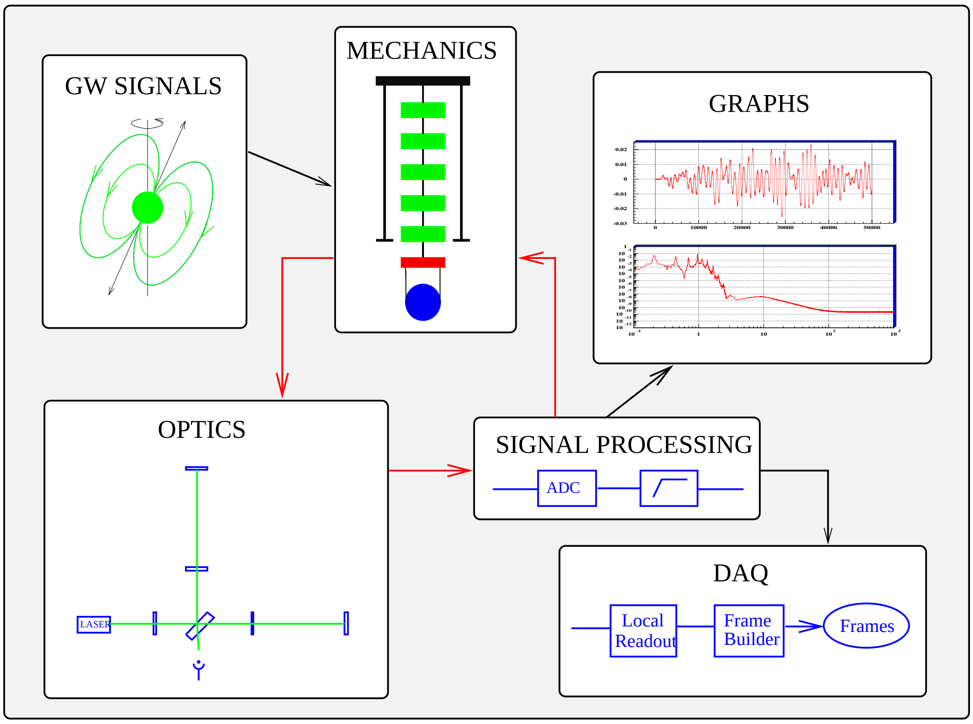

Figure 1: Pictorial view of the different topics implemented in SIESTA

Since the simulation works in the time domain, the heart of a job is basically a main loop running over time. And since it is object oriented, the configuration to simulate is not known a priori, but defined by the user by selecting the desired objects. At each iteration of the time loop, the engine core calls a sequence of routines associated with the different objects involved. These objects are defined in a text file called a "card file", which is written by the user and is therefore the program user input interface.

Three different entities can thus be distinguished in the software organization: the engine core, driving the main loop, the card file, providing the user interface, and the different modules (or libraries) housing the code associated with the various objects which may be used in the simulation. Let us now describe in more details each of these entities, as well as the program outputs.

2.2 Structure of a Simulation Object

The SIESTA modules contain the code associated with the different objects. The objects are the basic simulation units, each of them reproducing – in a reduced and simplified way – the functionality of some piece of hardware (or software). To mention only a few typical examples, objects include suspensions, mirrors, photo-diodes (the functionality of a photo-diode object being for instance that of a real photo-diode and its associated electronics, including a demodulator), analog to digital converters or digital filters.

Each object[2] is represented by a C structure and some associated functions acting on the C structure, called by the simulation engine core. Besides the standard constructors and destructors, those functions are the functions called during the initialization step of the simulation (initialization function), at each iteration of the simulation loop (simulation function), and at the end of the simulation job (termination function).

The various objects are gathered in different modules (or libraries) in a thematic way. A module contains all the objects related to some topic and can thus be thought of as a tool-kit for a given problem type, such as mechanics, optics or digital processing. Technically, a module is split into a header file containing the structure definitions and the function prototypes, and a source file containing the associated functions.

For all the objects, the fields of the associated C structure are split in two categories: the fields corresponding to input parameters, i.e. parameters whose values are user defined, and fields corresponding to internal or output parameters, whose values are set by the functions associated to the object.

[2]Strictly speaking, we should be talking here about structures – or classes – rather than objects. However the word "object" is usually used in SIESTA both for a structure and an instance of this structure, and we stick to this habit here.

Besides most objects representing concrete things, some special objects are used to exchange information between the other objects. Those objects acting as interfaces are more abstract: typical examples are light beams (used in optics) and signals, i.e. floating point values (used everywhere else).

To illustrate the latter case, the signal object is itself a very simple object, which is represented by a C structure having essentially two fields: a name field, and a data field containing the value of the signal, updated when necessary. The signal object is generic insofar as it can carry the information relative to very different things such as the position of a mirror in one degree of freedom, the output of a random number generator, the input signal of a digital filter or the output signal of a photo-diode electronics.

2.3 Structure of the Card File

The card file is a plain text file with a very simple syntax (see example in figure 2). Each entry in this file has the form of a keyword followed by some parameter values. The keyword has to match the name of a structure associated to one of the simulation objects. The parameter values following the keyword have to match the number, the order and the types of the input parameters of this structure.

The reading and parsing of the card file are done by the simulation engine core. Each time a keyword is read and recognized, a special function associated to the corresponding object – the function called constructor in the previous section – is called. This function is in fact more than a simple constructor, since its functionality is to allocate the memory necessary to store the object, to read the parameter values in the card file, to initialize the object fields with those values (after calling member constructors if necessary), and finally to connect the object to the main simulation loop (see section 2.4).

As mentioned in section 2.2, some special objects such as signals or light beams are used to exchange information between objects. A practical consequence of this is that the output signal of an object (object 1 – for instance a random number generator) can be the input signal of another object (object 2 – for instance a digital filter). In the parameter list of object 2, the input signal is referred to by its name, the name of the output signal of object 1 being known by convention from the name of object 1. The engine core takes care of the reference management implied by this mnemonics system, in order to assign correct values to pointer fields in all concerned objects once all the signals (and similar interface objects) are known, which occurs only when the card file has been parsed and all constructors have been called.

The first entry in the card file is related to a special object – the “clock system” – defining the parameters of the simulation main and secondary loops. As will be explained in greater detail in section 2.4, a simulation time loop is associated to each clock, the time step of the loop being the inverse of the clock frequency.

The order of the entries in the card file matters inasmuch as it determines the sequence of call-backs occurring in the simulation loop, thus defining the chronology of the different simulation actions.

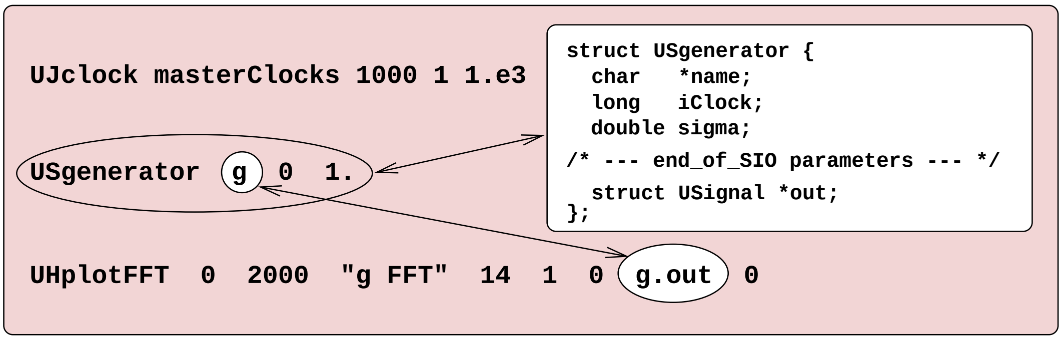

Figure 2: Example of the structure definition corresponding to a SIESTA object (left) and of a card file (right). This example illustrates the relationship between the structure fields and the parameters of the card file. Also illustrated is the use of mnemonics to exchange information between two objects.

2.4 Engine Core

The main simulation loop is a loop over time. This main loop includes a nested loop over the different objects involved in the simulation job. In fact, the latter loop runs over generic objects, called "clocking objects". Using the C++ analogy, this object type can be thought of as an abstract class from which any simulation object is derived and whose virtual functions are the initialization, simulation and termination functions defined for each derived object.

Since standard C is used, the code to support such an organization had to be developed. It is based upon a dictionary like system, where each valid simulation object is declared as such and where function pointers are used to reference the associated methods.

As mentioned in section 2.3, for every entry in the card file an instance of the corresponding object is created and a clocking object is scheduled in the simulation loop. During the initialization step, the engine core loops over all clocking objects to call the corresponding initialization functions, and the same process is used at each iteration of the time loop to call the simulation functions and at the end of the simulation to call the termination functions.

This scheme is actually somewhat complicated due to the fact that objects do not necessarily all share the same simulation time step, which technically translates into the objects not being all connected to the same clock. As a result, the list of objects for which a simulation routine is called may vary from one iteration of the main loop to another.

Besides this scheduling task, the engine core also takes care of the reference management needed to use mnemonics in the card file, as already mentioned in section 2.3.

Figure 3 summarizes the main features of the program structure and the relationships between the different entities.

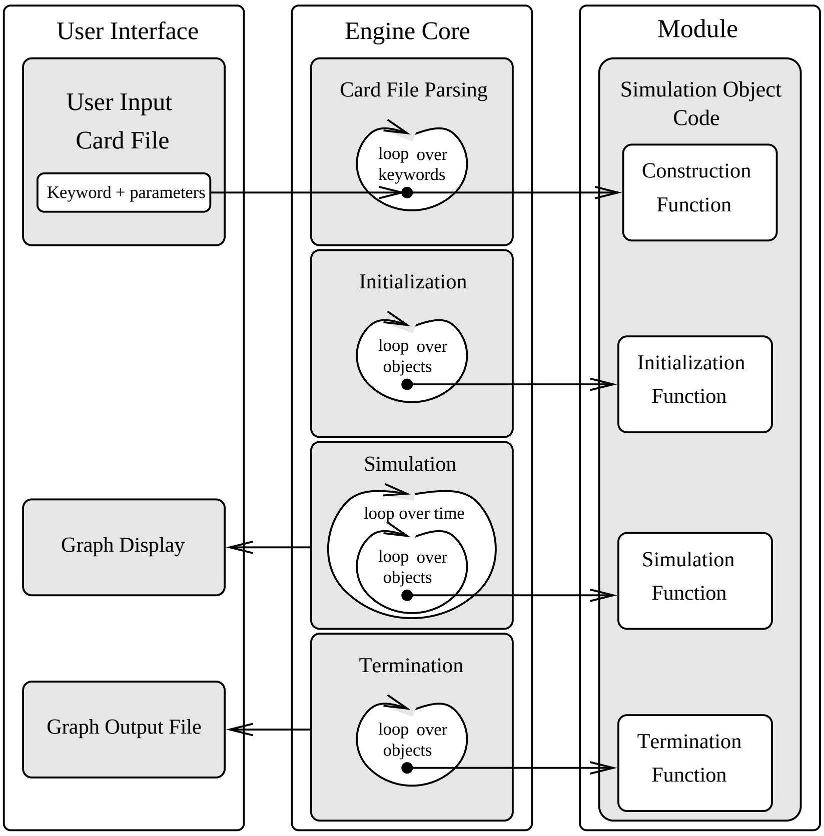

Figure 3: Schematic view of the SIESTA technical organization.

2.5 Program Outputs

The program outputs are:

- a listing file, which keeps track of the job conditions and of its progress from the technical point of view. This includes version numbers for the different modules, a copy of the card file, information about the parsing of the card file and about the reference management process, any warning or error report, and debug information if a high debug level has been selected for the job.

- one or more files containing output graphs about quantities selected by the user in the card file. SIESTA is interfaced to the HBOOK package from the CERN library [3] so that graphs with this format are created and output in a file which can be analyzed through the PAW software (also from the CERN library). It is also possible to have these graphs displayed on the screen during job execution. Since the CERN library is not quite available throughout the collaboration, it is also possible to dump the information about the selected quantities with a plain ASCII format in a file which can be subsequently read by any kind of graph viewer available to the user.

- a file containing formatted data including channels selected by the user. SIESTA contains objects simulating the data acquisition and is interfaced to the VIRGO/LIGO data formatting package [4] so that data can be written with the "frame" format just as real data are.

3 General Utility Modules

The general utility modules contain objects which do not simulate the functionality of any VIRGO sub-system (except for some objects simulating control or data acquisition software) but provide the tools needed to perform the necessary numerical computations or to produce output in a useful form. Hereafter we focus on the tools for signal processing, graph production and formatted data production. In addition to the latter, the general utility modules also include tools to perform basic linear algebra operations on matrices, as well as some simple mathematical operations such as χ^2 minimization.

3.1 Signal Processing Module

The signal processing module contains both very basic tools to generate signals of various types or to perform simple handling operations on signals, and actual signal processing tools such as digital filters. It turns out that a limited number of objects provide most of the tools generally needed.

- Besides simple objects designed to generate signals with usual shape such as sines or slopes, the generation tool-kit contains an object producing pseudo-random series with white and Gaussian distribution. These objects are commonly used to produce signals in order to excite other items. Especially the random number generator, used either alone or in combination with digital filters, can be used to produce white or colored noise.

- The handling tool-kit contains objects designed to perform basic operations on one or more signals, such as linear combination, delay or conditional assignment.

- The processing tool-kit contains a few general objects. Besides an object designed to simulate the action of an analog to digital converter by adding quantization noise to the input signal, the tool-kit contains essentially implementations of digital filters. The parameters assigned by the user are that of the analog filter the user wants to transform into a digital filter. In practice there are two different implementations of the analog to digital transformation [5], one based on impulse invariance and the other one based on the bilinear transformation. The choice of the implementation is up to the user, and is to be made on the basis of the analog filter characteristics.

3.2 Graph Producing Module

SIESTA is interfaced to the HBOOK package from the CERN library in order to produce graphs which can be displayed during job execution or analyzed afterwards using the PAW software. The graphs show information about relevant quantities of the simulation job that the user selects in the card file. Any signal object can be used to produce a graph, which means that the user can look at many internal data of the simulation. Technically, each graph to be produced is a separate entry in the card file, and therefore a separate object scheduled in the simulation. The information provided about the user selected quantities can be either in the time domain or in the frequency domain.

- **time domain**
  Since the simulation is performed in the time domain, this type of information is straightforward to extract from the data flowing through the different simulation objects. Objects producing graphs in the time domain take as input one or more signal objects providing the time sequences of the quantities of interest, and perform basic operations on those signal objects to extract the desired information: time evolution or distribution. There are also tools allowing to correlate two signals or more.

- **frequency domain**
  The information in the frequency domain about the quantities involved in the simulation is not directly available and needs dedicated tools to be extracted. These tools are very standard and make a heavy use of fast Fourier transform (FFT) algorithms. The information most commonly extracted in the frequency domain is the power density spectrum of a quantity, estimated from a sequence of values provided by a signal object. The estimation is based on periodograms computed by performing FFTs on time sequences, and uses standard techniques such as averaging, windowing, and whitening when necessary. There are also tools to extract the transfer function and the cross correlation between two signals.

3.3 Interface with Data Formatting Package

In order to be able to write simulated data with the same format as the real VIRGO data, SIESTA is interfaced to the VIRGO/LIGO data formatting package (the Frame Library) [4]. This interface consists essentially in objects simulating the functionality of part of the data acquisition software (local readout and frame builder). The interesting signal objects produced in the simulation job are "read-out" and packed into frames by these interface objects, using functions from the Frame Library.

Conversely, tools are implemented in SIESTA to read frame formatted data. This can be used to run data analysis algorithms implemented inside the SIESTA framework, either on simulated or real data. This possibility also allows to read real data in order to get realistic noise realizations, and add generated GW signals into them.

4 Mechanical Simulation Module

The purpose of the mechanical simulation module is to provide the tools to describe the random movement of the test masses, i.e. the suspended mirrors. This includes simulating external noise such as seismic noise, internal noise such as thermal noise, and of course the mechanical response of a mirror and its suspension.

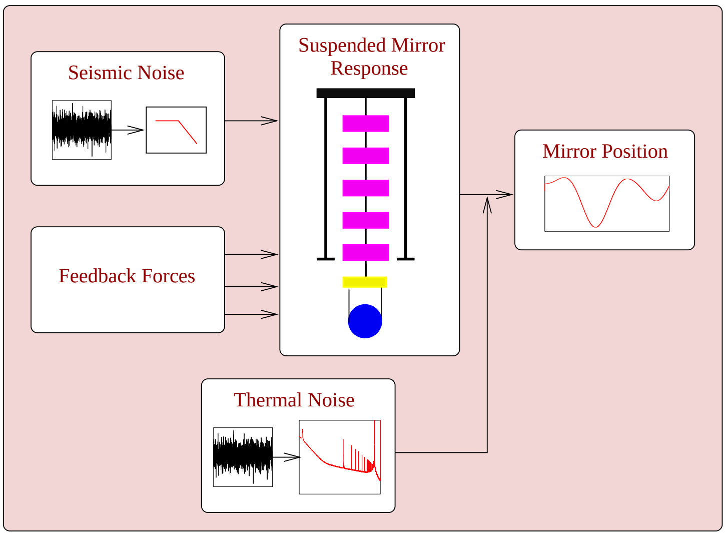

Figure 4: Schematic view of the organization of the mechanical simulation

Since SIESTA works in the time domain, simulating noise from any source means producing random time series with the proper frequency spectrum. Generally speaking, this is done by producing random series with a white spectrum, and filtering them in order to get colored noise with the expected spectrum. This is explained in more details in section 4.1.

Simulating the mechanical response of a suspended mirror means computing the position of the mirror as a function of time, taking into account various inputs - external noise, feedback forces applied at various levels of the suspension - and the behavior of the complex multi-pendulum structure. Several models are available to perform this task, with different detail levels. Early developments have concentrated on describing the mirror movement in one dimension and one degree of freedom, namely the mirror position along the light beam. On the other hand a recent and important development has led to a model able to describe the behavior of a mirror and its suspension in three dimensions and six degrees of freedom. The different models are reviewed in section 4.2.

Those general remarks are schematically summarized in figure 4.

4.1 Noise Simulation

4.1.1 Seismic Noise

The seismic noise generator simulates the displacement transmitted to the suspension through ground coupling. The generator produces random displacement in one dimension – along the light beam. In order to reproduce the frequency behavior observed experimentally, namely a white spectrum at low frequency and a spectrum in f^{-2} at higher frequency, the generator is based on filtering white and Gaussian noise with a second order low pass filter. The values for the amplitude of the input noise and the cut-off frequency of the low-pass filter are defined by the user to reproduce experimental measurements. Besides this simple built-in generator, white noise generation and signal filtering can be used by the user in order to produce a more detailed spectrum if needed.

A model generating seismic noise in more than one degree of freedom and taking into account realistic correlations between nearby ground points is yet to be developed, although not of crucial importance for sensitivity or global control studies.

4.1.2 Thermal Noise

The thermal noise generator is based on an effective model which does not provide a detailed description of the various physical processes leading to thermal noise but rather a description of the resulting displacement noise on the test mass. The contribution of each thermal noise source to the mirror position noise is modeled as [6]:

$$\tilde{x}_i^2(\omega) = \frac{4kTm_i\omega_i^2/Q_i}{\omega[(m_i\omega_i^2 - m_i\omega^2)^2 + m_i^2\omega_i^4/Q_i^2]} \tag{1}$$

where m_i, ω_i, Q_i are the effective mass, the angular resonant frequency and the quality factor of the mode.

This approach assumes that the parameters of this effective model can be extracted from other sources, as the results either from experimental measurements or from a modeling of the physical process involved [7].

The contributions of the different modes are then summed in order to compute the noise induced on the position of the test mass:

$$\tilde{x}^2(\omega) = \sum_{i=1}^{n} \tilde{x}_i^2(\omega)$$

For each mode, noise with the spectrum as in equation 1 is produced by filtering white noise with a series of two filters, a filter exhibiting a transfer function in $f^{-1/2}$, and a double-pole low-pass filter. The $f^{-1/2}$ filter itself is simulated by cascading first order filters with properly spaced zeros and poles[3] so that the transfer function approximates a $f^{-1/2}$ behavior over the frequency range of interest.

4.2 Suspended Mirror Simulation

Three models of the mirror suspension [1] are available, with very different detail levels. This reflects both the development chronology, with models of increasing complexity, and the variety of the practical needs.

Although generally speaking it is highly desirable to have a description of the suspension as complete and detailed as possible, it is often useful to have simpler and faster models. A detailed description in six degrees of freedom is needed if the purpose is to optimize the design of the suspension, develop the suspension control system, or study special aspects of the interferometer global control such as interaction between the angular and longitudinal controls of the mirrors. On the other hand, if only a simulation of the interferometer control in the longitudinal degree of freedom is required, a simpler, one dimension model of the suspension may be very useful to reduce the computational effort.

Let us describe briefly the three different models available:

- The simplest model of the suspended mirror is based on describing the multi-pendulum structure by a series of double-pole low-pass filters whose parameters – resonant frequencies and quality factors – are defined by the user. Despite the facts that this model is obviously extremely simplified, that it is a one dimension model and that it accepts a very limited number of inputs (seismic displacement of top point and feedback force applied directly on the mirror), it has some advantages that make it useful in some

[3]If p_i are the poles and z_i the zeros, we have $p_i \sim 3 \cdot z_i$ and $z_{i+1} \sim 10 \cdot z_i$

cases: it is fast and flexible – the number of stages being user defined – and experimental input can be easily injected in the model since measured values for resonant frequencies or quality factors can be used as parameters of the filters.

- A less simplified model of the suspended mirror but still accounting for the mirror movement in only one degree of freedom (the longitudinal position of the mirror along the beam) is based on solving the differential equation describing the movement of the multi-pendulum structure in one dimension:

  $$\mathbf{M} \cdot \ddot{\mathbf{X}} + \mathbf{\Gamma} \cdot \dot{\mathbf{X}} + \mathbf{U} \cdot \mathbf{X} = \mathbf{F} \tag{2}$$

  with $\mathbf{M}$ the mass matrix, $\mathbf{U}$ the potential matrix and $\mathbf{\Gamma}$ the dissipation matrix.

  The differential equation is re-written as a matrix state equation and solved numerically. This model takes more inputs and can account for forces applied on the marionetta or on the second stage [1], and has still some – limited – flexibility as the total number of stages can be modified to some extent. On the other hand, this model does not include the first filtering stage done by the inverted pendulum in the current design.

- The most detailed model presently available of the suspension system provides a simulation in six degrees of freedom [8]. It describes the suspension in its current design – which includes an inverted pendulum system – in terms of mass elements (filter bodies, payload components), elastic elements (flexible joints, suspension wires, blades, anti-springs) and active or sensing elements (coil-magnet pairs, accelerometers). There is some flexibility in the way elements are introduced and connected together, and are assigned numerical parameters: the module has been designed in order to be upgraded following the needs of VIRGO. The high detail level is obtained at the price of a higher computational effort, as the state space vector is enlarged roughly by a factor of ten. The model also lacks the possibility of adjusting the parameters to experimental measurements, for instance it is not possible in the current release to adjust the Q values of the resonances – although this limitation should be overcome in the future.

4.3 Performances

Let us give here some typical results about the performances of the mechanical simulation. The computation times indicated have been measured on a digital™ ALPHA 500/400 workstation[4].

[4] SPECint95 = 12.3 SPECfp95 = 14.1

Table 1 gives the computation times needed to simulate the mechanical response of a VIRGO-like mirror suspension with the three models described above, over one second with a simulation time step of 100 μs.

The large differences in the computation times reflect directly the differences in the detail level of the different models.

Model	Computation Time for 1 s @ 10 kHz
1 d.o.f. filter series	30 ms
1 d.o.f. differential equation resolution	90 ms
6 d.o.f. differential equation resolution	9 s

Table 1: Computation time needed to simulate with three different models the mechanical response of a VIRGO-like mirror suspension over one second, with a simulation time step of 100 μs.

5 Optical Simulation Module

The optical simulation module is rich of a number of models differing either in the configuration simulated or in the method used. The variety in the available models reflects the diversity of the needs as far as the optical simulation is concerned.

In addition to the standard, most often simulated configurations – Fabry-Perot cavity, recycled Michelson or recycled Michelson with Fabry-Perot cavities – it is useful to have the possibility to define other types of configuration from individual mirrors.

The detail level one wishes to consider varies also a lot from one application to another. In some cases for instance it is enough to simulate the response of the interferometer to longitudinal displacements of the mirrors, whereas in other cases the angular degrees of freedom must be taken into account. We can even be interested in the details of the transversal structure of the beam leaving the interferometer, which implies that the mismatches between the beams and the mirrors – either due to surface curvature or rugosity – have to be properly taken into account.

Similarly, there are some cases – for control purposes for instance – where one needs to follow dynamically the evolution in time of the interferometer response as a function of the mirror movements which can be fast enough – given the time constants involved in the different cavities – so that the response is not the stationary one but we are sensitive to transients. On the other hand it is often enough to follow the interferometer response under the quasi-static assumption. The needs happen to match the resources here, since the cases where a detailed description is required are also those where the quasi-static approximation is acceptable. This is fortunate since those detailed models are generally slow and would be difficult to implement in a dynamic mode.

Before reviewing the various models, let us say a few words about the technical aspects of the optical simulation.

5.1 Technical Aspects

The configuration to be simulated is defined by the user using a number of basic objects. These objects are connected to build a logical sequence. Typical objects are lasers, phase modulators, mirrors, cavities, interferometers or photodiodes. They are connected through "beam" objects, much in the same way other SIESTA objects are connected through signal objects. The beam objects are used to exchange information between one optical object and another, and contain the representation of the optical field in some place of the configuration. The representation itself may vary from a model to another, as will be explained in section 5.3.1, but the object structure is generic. Usually, both the input and the output of an optical object are beams, and the beams are updated by the simulation functions associated to the various objects.

The interface with other simulation modules such as the mechanical simulation one is done through special objects – namely mirror surfaces – whose parameters define both geometrical and optical properties (such as position, direction, reflectivity, losses and so on) and which are linked to signal objects carrying information about position variation of the mirrors (typically updated by mechanical simulation objects). Those surface objects are used as input parameters of objects such as cavities or interferometers.

5.2 The Laser Object

The laser object comes first in the definition of an optical configuration. Its function is to produce the initial beam to be propagated. The transverse mode composition of the laser beam can be defined by the user, and frequency sidebands can be generated in order to propagate a phase modulated beam.

The laser power and frequency noises can be taken into account through input signals to the laser object, which allows to inject noise with any desired spectrum, by using white noise generators together with signal filtering tools.

5.3 Light Propagation

The different models which have been developed can be classified as a function of the representation they use to describe the optical field, which is closely related to the algorithm they use to compute the field solution, and also to the kind of effects they are sensitive to.

Another important feature of the models is the approximation they use for the time evolution of the fields. Some of the models allow to compute only the stationary solution for the fields – and therefore can be used to describe their evolution in time only under the quasi-static assumption – whereas other models can also describe transients.

5.3.1 Field Representation

Hereafter we describe briefly the different field representations, their consequences on the algorithms used to solve for the field and the kind of effects they are able to sense.

- **plane wave:**

  This is the simplest field representation, since all the information about the field is held in a single complex number. Under this assumption, fields are propagated along distances between mirrors simply by scalar phase factors. This representation is well suited if one needs only to compute the configuration response as a function of the mirror positions along the beam axis.

- **grid/FFT:**

  In this representation the transversal structure of the field is discretized, and the field value is computed in a number of points equally spaced on a square (the "grid"). The field propagation between mirrors uses an algorithm based on FFT [9]. The reflection on a mirror is accounted for by computing the extra phase in each point of the grid, taking into account any small misalignment of the mirror, its curvature, and its rugosity if any. These are therefore the kind of things this representation is able to take into account.

- **modal expansion:**

  This type of representation is the most used one in SIESTA, and is so useful that it has triggered several developments and is now implemented in three different ways.

  In this representation the field propagating through the configuration is expanded on a basis of orthogonal modes. If the basis is chosen such that the basic modes are close to the eigenmodes of the configuration, then it is generally possible to truncate drastically the basis and consider only a small number of modes. In practice the Hermite-Gauss $\mathrm{TEM}_{\mathrm{mn}}$ modes are used, and the modes actually taken into account are such that $(\mathrm{m}+\mathrm{n})$ is less than some order which is usually not much larger than unity.

In this representation the information about the field at some place of the configuration is held in a vector whose elements are the field components on the basic modes. The elementary operations such as propagation or reflection are represented by operators acting on the field state vector. Free propagation and reflection by ideal mirror surfaces are represented by diagonal matrices, whereas reflection by misaligned or mismatched surfaces is represented by matrices with off-diagonal terms leading to coupling between the different modes.

Let us now review the three different implementations of this method:

- *standard:*

  In the standard implementation the reflection matrices for not ideally aligned mirror surfaces are computed using analytical formulas given at order 2 in the relevant perturbation parameter [10]. The perturbation can be the reflection by a tilted plane or spherical mirror, reflection by a transversally displaced spherical mirror or reflection by a spherical mirror whose curvature does not match the beam curvature. The order of the truncated $\mathrm{TEM}_{\mathrm{mn}}$ basis is upper limited to 2 $(\mathrm{m}+\mathrm{n} \leq 2)$ but can be chosen by the user between the values 0 (in which case the model is merely equivalent to a plane wave calculation), 1 and 2.

- *special:*

  The special implementation is very similar to the standard one, except that a dedicated package has been developed to optimize the numerous algebra operations by taking into account the characteristic structure of the reflection matrices [11].

- *extended:*

  This last implementation uses a numerical method – instead of analytical formulas – to compute the reflection operators [12]. The consequences are that the order of the $\mathrm{TEM}_{\mathrm{mn}}$ basis is not limited (although in practice limitation arises from the finite computer memory resources), that the matrix elements are computed with better accuracy, and that mirror surface rugosity may be taken into account.

  The method used to compute a mirror reflection matrix is the following. For each mode of the basis considered as an incoming field at the level of the mirror tangent plane, a discrete spatial representation of the field transversal distribution is computed on a $n \times n$ points grid (with typically $n = 128$). The phase introduced by the reflection on the mirror is computed in each point of the grid taking into account the curvature of the mirror, any small misalignment, and the surface rugosity if any. The numerical projection of the reflected field on each mode of the basis leads to the elements of one column of the reflection matrix. Repeating this process for every mode of the basis provides the whole matrix.

5.3.2 Field Equation Resolution

The equation for the field in some place of the configuration is typically an implicit equation, which can be solved either by iterations or by direct inversion. The two methods are used in SIESTA, and the choice of the method generally determines whether the model is dynamic or quasi-static. Direct inversion of the equation implies that the field is solved for its stationary solution and therefore that the model is able to describe its evolution only in quasi-static mode. On the other hand, solving the equation by iterations means that field values at a given time step are computed from field values at the previous time step and therefore that the model is able to describe the field evolution dynamically.

The latter statement has to be moderated due to the fact that it is true only if the time step can be chosen in accordance with the physical delay introduced by the propagation of light. Depending on the field representation used, this can be so time consuming that in practice the model can only be used in quasi-static mode. This is especially the case of the grid/FFT model which is implemented with the sequential method but is much too slow to be used dynamically.

Actually there are also some cases where both methods are used in conjunction. This is the case for one of the most powerful implementations of the simulation of a recycled Michelson interferometer with Fabry-Perot cavities. In this model, the fields reflected by the Fabry-Perot cavities are computed sequentially whereas the field inside the recycling cavity is computed by direct inversion. This is equivalent to neglecting the time delays introduced by the light traveling inside the recycling cavity. Since the length ratio between the Fabry-Perot cavities and the recycling cavity is so high for an interferometer like VIRGO, this does not preclude the model from reproducing with good accuracy the time constants involved in such a configuration and therefore from describing properly the dynamics of the system.

It seems worth pointing out that in most cases (at least for those models implemented in the dynamic mode) the time step of the simulation is set by the frequency of the clock associated to the corresponding object. This means that the user is in a position to choose the time step suitable for his needs.

5.4 Optical Configurations

A variety of objects are available to simulate the optical response of different configurations.

- It is often useful to be able to define the desired configuration modularly from individual mirrors. An object called "optical node" provides the means to do so, the configuration being defined by connecting such node objects – representing mirror surfaces – with beam objects.
- Some configurations are so often considered that the models to simulate their responses have been encapsulated in dedicated objects, for which the geometry is fixed, most other parameters being user defined. This is the case for:
  - a Fabry-Perot (FP) cavity
  - the configuration of the VIRGO central interferometer, namely a recycled Michelson interferometer
  - the configuration of the full VIRGO interferometer, namely a recycled Michelson interferometer with Fabry-Perot cavities

Table 2 summarizes the methods used by the models available to simulate the various configurations – without taking into account the fact that a model dedicated to a complex configuration can be used to simulate a simpler one by adjusting some parameters.

		FP cavity	Recycled Michelson	Recycled Michelson with FP cavities	Modular Configuration
plane wave				QS	
modal expansion	standard	QS & D	QS	D	D
	special			QS	
	extended		QS	QS	
grid/FFT					QS

Table 2: Summary of the methods implemented for different configurations (a model dedicated to a complex configuration can of course be used to simulate a simpler one by adjusting some parameters). The table also indicates whether the method is implemented in quasi-static (QS) or dynamic (D) mode.

5.5 Performances

In this section we briefly present some results about the typical performances of the optical simulation, as measured on the same workstation as in section 4.3. We give two different bunches of results, one for the quasi-static models, and one for the dynamic ones.

5.5.1 Quasi-static Models

Table 3 refers to the quasi-static models. It shows the computation time needed to calculate once the stationary response (i.e. the stationary solution for the field in a number of places of the configuration) of three different configurations using different methods. The number of frequency bands considered is three – a carrier and two sidebands. The order indicated for the methods based on modal expansion refers to the maximum value of $(\mathrm{m}+\mathrm{n})$ of the $\mathrm{TEM}_{\mathrm{mn}}$ modes considered. If the order is p the total number of modes is $(p+1)(p+2)/2$. The computation times given in table 3 do not include overhead, which amounts to 0.4s.

The test with the grid/FFT method was done with a complete interferometer configuration built modularly. The grid size was 128×128 points. With this method the fields are computed sequentially, and the number of iterations performed was that needed to reach the stationary state of an interferometer with a cavity finesse of 50 and a recycling factor around 50, starting from fields initialized to zero. Convergence could of course be reached much faster starting with suitably initialized fields or by using fast iteration techniques [13]. Let us also emphasize that the grid/FFT method was an early development in SIESTA and that by now the modal expansion method in its extended implementation can provide the same kind of information at a much reduced computational cost, although for dedicated configurations only.

			FP cavity	Recycled Michelson	Recycled Michelson with FP cavities
plane wave					4 ms
modal expansion	standard	order 0	0.15 ms	0.01 ms	
modal expansion	standard	order 1	0.3 ms	0.3 ms	
modal expansion	standard	order 2	0.9 ms	1.4 ms	
modal expansion	special	order 2			24 ms
modal expansion	extended	order 5		1.5 s	
modal expansion	extended	order 10			25 s
grid/FFT					20 hours

Table 3: Computation time needed to calculate the stationary optical response of various configurations with different quasi-static models.

5.5.2 Dynamic Models

Table 4 refers to the dynamic models. It shows the computation time needed to simulate the response of two different configurations over one second, using different methods (again for three frequency bands and neglecting overhead).

The frequency given in the table (10 kHz, 100 kHz, 25 MHz) is the frequency with which the simulation function of the model is called, thus defining the simulation time step.

The length of the Fabry-Perot cavities considered in the tests is 3 km. To simulate dynamically the response of such a cavity, the basic time step of the sequential field computation must be at most 10 μs. Whether the call-back frequency is 100 kHz or 10 kHz, the time step is exactly 10 μs in both cases, the fields in the recycling cavity being computed through direct inversion. This means that in the 10 kHz case, several iterations are performed within the same function call. This allows to reduce overhead, at the price of simulating the cavity response up to 10 kHz only, which is often enough given the typical velocities of the suspended mirrors.

On the other hand, we give also the computation time needed in an extreme case (the 25 MHz one). In this case all the fields in the interferometer, including those in the Fabry-Perot cavities, are computed sequentially with a time step of 40 ns, corresponding to the propagation time of light in a VIRGO like 12 m long recycling cavity.

			FP cavity	Recycled Michelson with FP cavities
modal expansion (standard implementation)	10 kHz	order 0	0.7 s	0.4 s
		order 1	2.4 s	18 s
		order 2	8.2 s	89 s
	100 kHz	order 0	2.7 s	1.4 s
		order 1	12 s	47 s
		order 2	50 s	195 s
	25 MHz	order 0		40 min

Table 4: Computation time needed to perform a "one second" simulation of the optical response of two configurations, using a dynamic model based on modal expansions of different order and different simulation time steps.

6 Basic Tools for Data Analysis

The basic tools to produce simulated data are implemented in SIESTA. They include generators of gravitational wave signals, a noise generator, as well as the interface to the frame formatting package. The latter has already been described in section 3.3 and we focus here on the first two points.

It is to be noted that the SIESTA framework allows also to implement and run any data analysis algorithm as a SIESTA object, the data being processed in the standard SIESTA loop.

6.1 GW Signal Generators

The implementation of GW signal generators has two aspects. One aspect is the computation of the GW signal amplitude in the source frame, based on parameters depending on the model used. The other aspect is the interface to the detector frame in order to compute the signal the interferometer is sensitive to. Implemented in SIESTA so far are the source-detector interface and two basic generators at source level – one for continuous waves from pulsars [14], and one for coalescing binaries [16] (up to the first post-Newtonian order).

The source-detector interface is implemented as a SIESTA object transforming any signal in its source frame into a signal in the detector frame [14]. Namely, knowing the source position, the wave polarization angle and the detector position and orientation as a function of time, this object computes the values of h_x and h_y at detector level as a function of time from the generated values of h_+ and $h_\times$ at source level. The key input to this transformation is the Earth position as a function of time, which is computed with a routine provided by the Bureau des Longitudes in Paris [15].

This step introduces the amplitude and phase modulation of the GW signal due to the variations in the detector position and orientation. Technically, only the amplitude modulation is introduced through this general mechanism. On the other hand the issue of phase modulation is addressed at the level of the source, for reasons of convenience. It is indeed much simpler to account for this effect through an additional varying phase in the generated signal than through delayed arrival times at the detector level.

6.2 Noise Generator

For the purpose of developing and testing data analysis algorithms, it is often enough to work with time realizations of detector noise in terms of h (i.e. reconstructed data) with a frequency content reproducing the detector sensitivity.

It is possible – although slow and heavy – to combine GW signal generation with a full simulation of the detector (including mechanics, optics, controls). This produces raw data (dark fringe signal) and implies that a reconstruction procedure is applied to unfold the response of the servo-ed interferometer and get data in terms of h.

On the other hand, a lighter solution consists in producing directly time series in terms of h. The noise generator implemented in SIESTA is based on this idea and is designed as a fast tool to produce colored noise with the desired spectrum. The contribution of shot noise to the spectral sensitivity is simulated from a few parameters (power on beam-splitter, cavity length and finesse). The contribution from thermal noise is taken into account through an interface to the thermal noise generator described in section 4.1.2, which avoids to duplicate code.

Some important noise contributions are missing in the generator. Those missing sources are less well known and difficult to predict; they include noise from electronics, controls, and generally speaking all sources of so-called excess noise. It is likely that an ad-hoc simulation of those noise contributions can be implemented only when the detector has been built and characterized.

Figure 5 shows the spectral density of a simulated time series of data in terms of the reconstructed main signal h.

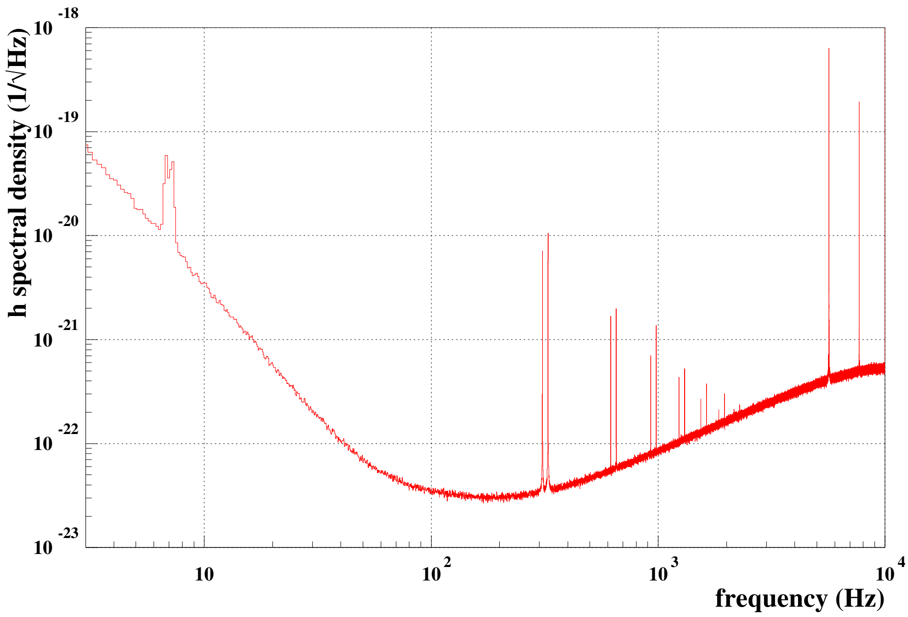

Figure 5: Spectral density of simulated noise in terms of reconstructed main signal h, including shot noise and the main contributions from thermal noise: pendular modes, violin modes, and the first internal modes. The time series was generated with the fast noise generator.

7 Application Examples

In this section we present a few examples to illustrate some of the typical applications of the SIESTA software.

7.1 Longitudinal Lock Acquisition of Fabry-Perot Cavity

This example (see figure 6) shows how SIESTA can be used to simulate the process of longitudinal lock acquisition. The configuration considered here is that of a simple suspended Fabry-Perot cavity. A dynamic optical simulation of the cavity is used in connection with a one degree of freedom mechanical simulation of the seismically excited suspended mirrors. Some signal processing tools are used to implement the lock acquisition algorithm and the feedback loop. Some graphs showing the time evolution of the process are produced in the SIESTA job by spying the relevant variables. Around 30 SIESTA objects are involved in this example which takes a little more than 6 minutes to simulate the evolution of the configuration for roughly 2 minutes.

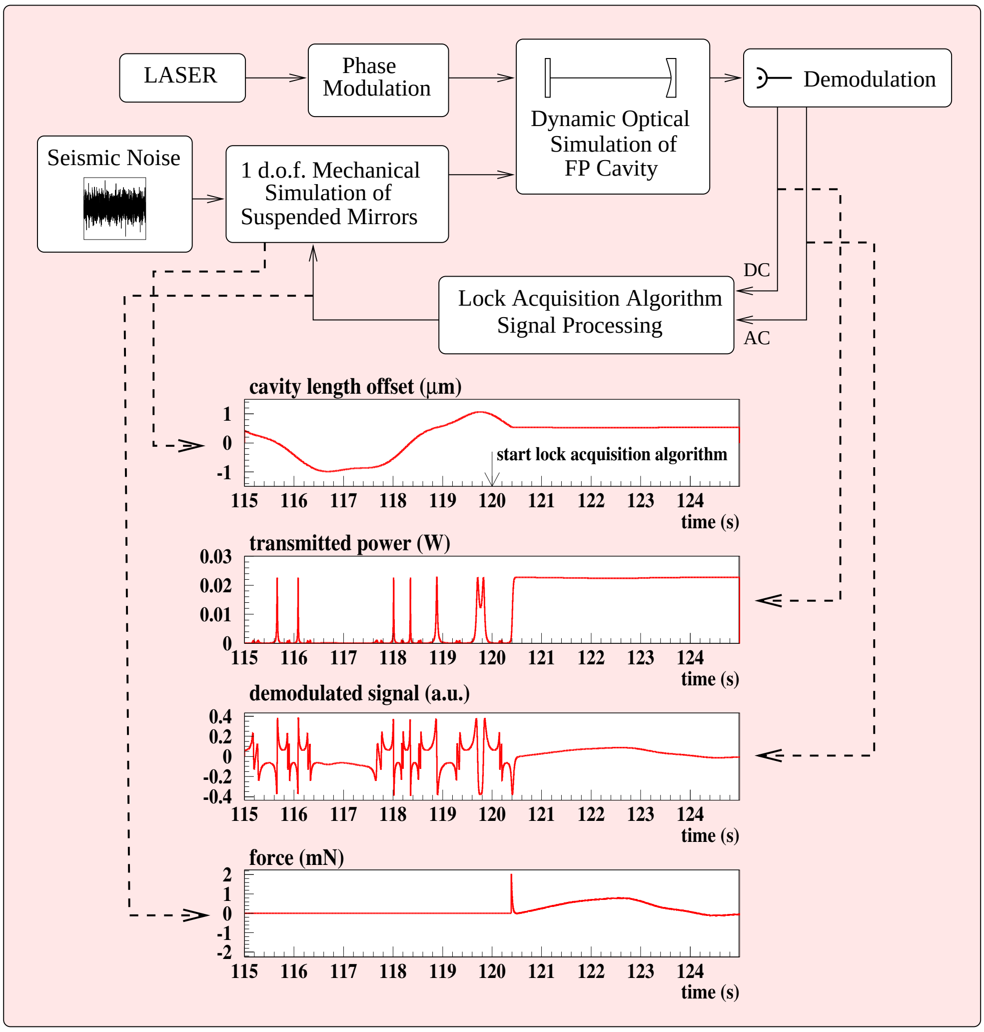

Figure 6: Schematic description of the ingredients and outputs of a SIESTA job devoted to the study of the longitudinal lock acquisition of a suspended Fabry-Perot cavity.

7.2 Transfer Function Extraction

This example (see figure 7) shows how SIESTA can be used to extract information in the frequency domain. We consider here the case of the extraction of the transfer function between the length variations of the VIRGO interferometer recycling cavity and the resulting variations in the demodulated signal measured on the photo-diode collecting the light reflected from the interferometer. A dynamic optical simulation of the interferometer is used with white noise applied on the longitudinal position of the recycling mirror. The power seen by the photo-diode at the modulation frequency is then evaluated and the transfer function with respect to the excitation noise is computed.

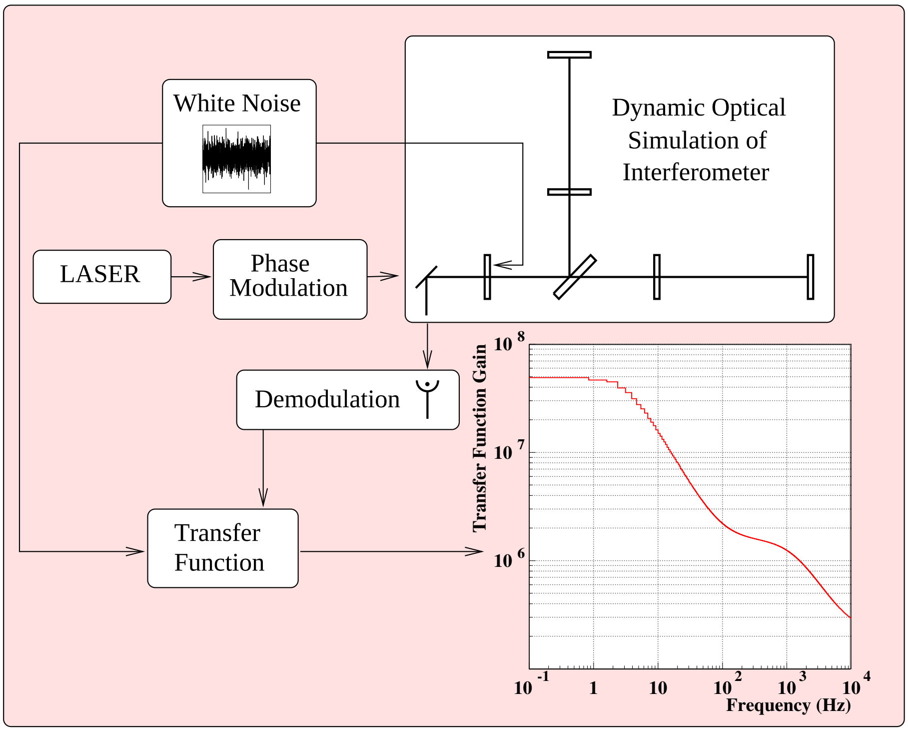

Figure 7: Schematic description of the ingredients and outputs of a SIESTA job devoted to the extraction of the transfer function between the length of the interferometer recycling cavity and the demodulated signal of the photo-diode collecting the light reflected from the interferometer.

7.3 Extraction of Alignment Error Signals

This example (see figure 8) shows how SIESTA can be used to simulate the alignment error signals produced by the seismic angular noise of the suspended mirrors. The configuration considered here is that of a recycled Michelson interferometer (VIRGO central interferometer). A quasi-static optical simulation of the interferometer is used in connection with a six degree of freedom mechanical simulation of the suspended mirrors. Assuming perfect longitudinal locking, the signals measured at the modulation frequency with a quadrant photo-diode collecting the light transmitted at the end of one of the interferometer arms are computed. Graphs show the time evolution of the up-down asymmetry of these signals, and its correlation to the vertical tilt of the recycling mirror.

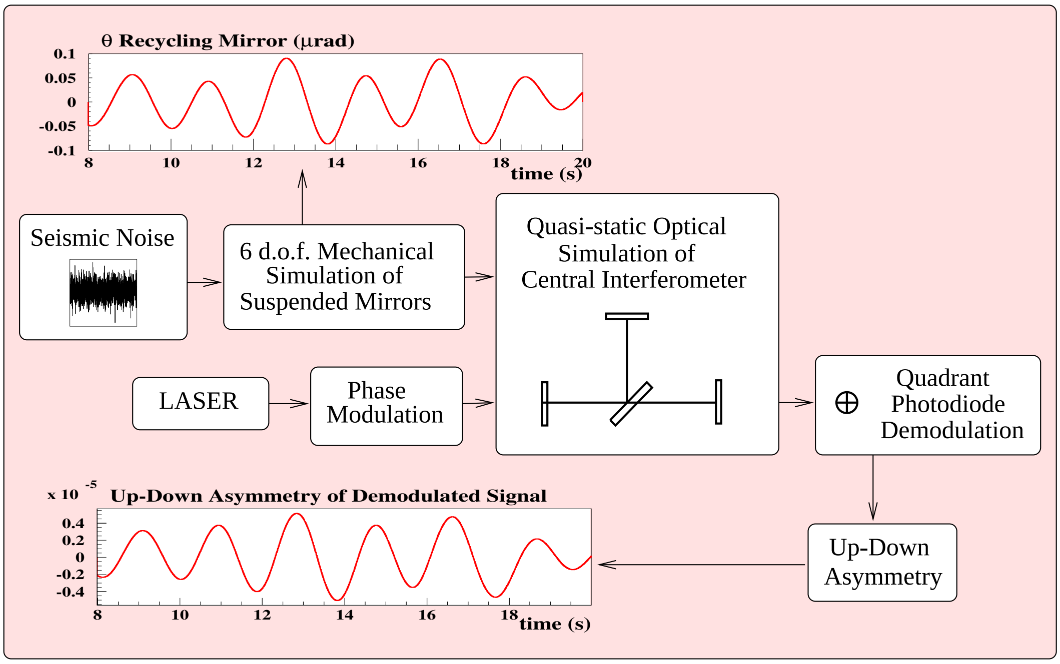

Figure 8: Schematic description of the ingredients and outputs of a SIESTA job devoted to the simulation of the alignment error signals produced by the seismically induced mirror angular noise and measured with a quadrant photo-diode collecting the light transmitted at the end of one of the central interferometer arms.

8 Conclusion

In this paper we have presented the SIESTA simulation software package developed for the VIRGO experiment. Stress has been laid on the modularity of the program. This feature makes it possible for instance to easily include new modules or re-use part of the code in other VIRGO software packages used online. Another important point is the possibility for the user to choose in many cases between several detail levels to simulate a given process.

It is worth reminding that many other modeling developments have been undertaken for the VIRGO experiment, which have not been mentioned in this paper devoted to the global simulation of the detector. In some cases these other modeling activities address very specific issues which need not be integrated in the general simulation. In other cases they are relevant to important physical processes characterizing the detector but only their results are implemented in SIESTA as an effective model. An important illustration is the simulation of thermal noise.

The development of SIESTA has been going on for several years [17] and has been driven by the needs of the VIRGO collaboration. The program has reached a stage where the design requirements have been fulfilled to a large extent, and is now at an operational level. One of the early needs has been to develop tools to help in the design of the detector. An important design issue still existing today is that of the control of the interferometer, which has driven most of the developments relative to optical and mechanical simulation.

The optical simulation toolkit is fairly complete by now. It allows to simulate the response of various optical configurations to longitudinal and angular movements of the mirrors, either in the quasi-static or in the dynamic regime. Future activities in this domain might be to develop efficient tools to simulate new interferometer configurations, like for instance dual recycling.

The mechanical simulation toolkit covers also the essential needs, allowing to simulate a suspended mirror in six degrees of freedom. Future developments will be needed to introduce more flexibility in the simulation. Another challenge would be to increase the speed of the mechanical simulation by optimizing the models.

The other main development line of SIESTA is directed towards data analysis. It concerns generators of gravitational wave signals as well as generators of noise reproducing the sensitivity of the detector. At present the basic tools have been developed, but further developments will be needed to extend the variety of signal generators and bring the noise generators closer to the noise observed in reality. In particular excess noise, including non-Gaussian noise, will have to be modeled or parameterized when it has been observed and characterized on the real detector.

Generally speaking, the program will have to be compared with reality like the coming up VIRGO central interferometer, and its parameters should be tuned. Future developments might also be desirable in order to introduce more interactivity in the program and to connect it at best with the data analysis software environment.

Acknowledgments

We would like to thank our VIRGO colleagues for many profitable discussions, and especially all the SIESTA users for useful comments and suggestions.

References

[1] VIRGO Coll., Final Design Report (1997)

[2] B.Bhawal, M.Evans, E.Maros, M.Rahman, H.Yamamoto, LIGO note LIGO-T970193 (1997)

[3] See http://wwwinfo.cern.ch/asd/index.html

[4] B.Mours, Gravitational Wave Detection, Proceedings of the TAMA International Workshop on Gravitational Wave Detection, Saitama, Japan, November 12-14 1996, edited by K.Tsubono, M.K.Fujimoto and K.Kuroda (1997) 27
Joint LIGO/VIRGO note LIGO-T970130-B and VIRGO-SPE-LAP-5400-102

[5] A.V.Oppenheim, R.W.Schaffer, *Digital Signal Processing*, Prentice-Hall International Editions (1975)

[6] P.R.Saulson, *Phys. Rev. D* **42** (1990) 2437

[7] G.Cagnoli, L.Gammaitoni, J.Kovalik, F.Marchesoni, M.Punturo, VIRGO note VIR-NOT-PER-1390-84 (1997), and references therein

[8] M.Beccaria, G.Cella, G.Curci, A.Viceré, VIRGO note in preparation

[9] J.Y.Vinet, P.Hello, C.N.Man, A.Brillet, *J. Phys. I France* **2** (1992) 1287

[10] J.Y.Vinet, VIRGO note PJT-94-012 (1994)

[11] J.Y.Vinet, F.Cavalier, P.Hello VIRGO note NTS-95-032 (1995)

[12] V.Loriette, VIRGO note NTS-95-029 (1995)

[13] B.Bochner, Ph.D thesis, Massachusetts Institute of Technology (1998)

[14] X.Grave, Thèse de l'Université de Paris-Sud (1997)

[15] G.Francou, L.Bergeal, J.Chapront, B.Morando, *Astron. Astrophys.* **128** (1983) 124

[16] D.Verkindt, Thèse de l'Université de Savoie (1993)

[17] B. Mours, I. Wingerter-Seez, Proc. of the sixth Marcel Grossmann Meeting on General Relativity, Kyoto, June 1991, edited by S.Humitaka and T.Nakamura (1992) 1576
B.Caron et al, Proceedings of the 6th Pisa Meeting on Advanced Detectors, La Biodola, Isola d'Elba, Italy, 22 - 28 May 1994, edited by E Bertolucci, F Cervelli and A Scribano, *Nucl. Instrum. Methods Phys. Res.*, **A : 360 (1-2)** (1995), 375
VIRGO Coll., Gravitational Wave Detection, Proceedings of the TAMA International Workshop on Gravitational Wave Detection, Saitama, Japan, November 12-14 1996, edited by K.Tsubono, M.K.Fujimoto and K.Kuroda (1997) 21