

# Integrated modeling tools for large ground based optical telescopes

George Angeli<sup>a</sup>, Anna Segurson<sup>a</sup>, Robert Upton<sup>a</sup>, Brooke Gregory<sup>b</sup> and Myung Cho<sup>a</sup>

<sup>a</sup> New Initiatives Office, AURA Inc., Tucson, AZ, USA

<sup>b</sup> Cerro Tololo Inter-American Observatory, NOAO, La Serena, Chile

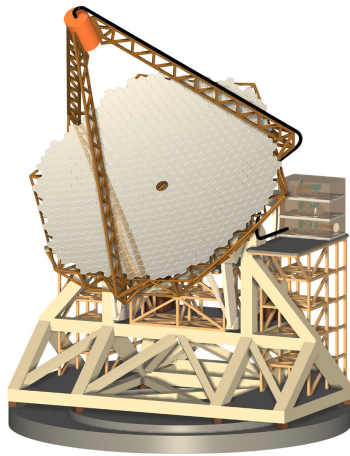
## ABSTRACT

Diffraction-limited performance of 30-m class telescopes requires the integration of structural, optical and control systems to sense and counteract real time disturbances to the telescope. Accurate simulation of an integrated telescope model is essential for optical performance estimation and design validation. Our approach to integrated time domain modeling of large telescopes is to interface commercially available structural, optical and control modeling software packages. The model architecture, data structures, and the interfacing tools of the simulation environment are presented. Preliminary simulation results of a 30-m class telescope subject to wind load and a ground layer phase screen are presented.

**Keywords:** Active optics, telescope control, telescope performance prediction, integrated modeling

## 1. INTRODUCTION

Achieving diffraction-limited imaging with the next generation of giant telescopes poses many challenges, and the telescope design is made more difficult by the need to keep costs significantly lower than would be predicted by scaling the cost of current eight to ten meter designs.<sup>1</sup> To simultaneously achieve cost and performance goals, all components of the telescope must be developed as part of an integrated system. The major tools supporting the design and realization of a Thirty Meter Telescope as a system are an *integrated computer model* capable of predicting the performance of the telescope, a *parametric cost estimate* capable of predicting the cost of the telescope, and a *science merit function* capable of linking the engineering performance and the cost of the telescope to the scientific goals and preferences of the relevant astronomical community.



**Figure 1.** Point design of the Giant Segmented Mirror Telescope.

---

Further author information (Send correspondence to G.A.):

G.A: E-mail: gangeli@gemini.edu; Address: AURA New Initiatives Office, 950 N. Cherry Ave., Tucson, AZ 85719, USA.

In the system engineering of a complex telescope, an integrated telescope model plays a key role in verifying the interactions and interfaces between various areas of the design (structure, optics and control). A pre-eminent example is the design and verification of the numerous control loops necessary for the appropriate operation of the Thirty Meter Telescope, including the considerations of structure-control and loop-to-loop interactions.

The model reported in this paper is developed from the point design of the Giant Segmented Mirror Telescope (GSMT)<sup>2</sup> (Figure 1); however, the tools used in and improved for this model can be applied to other similar telescopes.

From a project management and system engineering standpoint, there are two major requirements for an integrated model:

- The model must be transparently **verifiable** with the least possible effort. Because fundamental programmatic and technical decisions are supposed to be based on the simulation results, the consequences of an error can be tremendous. A lack of credibility may result in double and triple checking of every conclusion reached by using the model, which negates the purpose of the model.
- The model must be **flexible**; the use of the tools to build and especially to modify the model should require the least possible effort from the project team. Even the most beautiful and credible tools can become irrelevant in the budget- and time-critical environment of a real project, if maintaining those tools requires significant resources that extend beyond the typical project team.

During the last decade, several tools were developed for integrated modeling of large optical systems.<sup>3-6</sup> A major driver of these developments was the expanding technology of space telescopes. Practically all of these tool sets feature custom optical kernels, ranging from simple ray-tracing (IMOS),<sup>7</sup> to sophisticated multi-tool bundles (BeamWarrior<sup>4</sup>) that include Gaussian decomposition. Another common characteristic is the use of MATLAB as the integrating platform.

Unfortunately, the engineering and scientific community has very limited access to these cutting edge software packages. Some packages are clearly proprietary and not for sale, most of them are prohibitively expensive, and almost none of them release the source code. We need a package that allows the user to write routines in high level languages to facilitate user customization of the tool for specific projects.

Our approach to providing an integrated environment to the telescope design team is based on the requirements listed above. Instead of developing custom tools for representing the structure and especially the optics, we focused on efficiently linking together commercially-available and widely-used software packages. Therefore, the various components of the integrated model can be pulled from the software tools the different trade groups (optical, structural, and control engineering) are using for the actual design. With minimal modification these components can be used in system level studies. We kept the MATLAB/Simulink environment as an integration platform because this is the most comprehensive dynamic simulation environment available.

## 2. OVERVIEW OF THE MODEL

The major challenge of using commercial software packages is the need for synchronizing the structural and optical model. The problem is far from trivial, because the two models reside in different packages featuring dissimilar software tools. To solve the problem, we invoke the concept of a “virtual telescope” first introduced in the 1970’s to support the increasing complexity of telescope pointing control software.<sup>8</sup> In our extension, the virtual telescope is an encapsulated algorithm simulating some independent aspect of the telescope. In this sense we can talk about a virtual telescope in the structural domain and a virtual copy of the same telescope in the optical domain. The task is to update the prescription of the optical virtual telescope to match the changes made in the structural domain.

The structural model, as a dynamic virtual telescope, is built in MATLAB as a set of differential equations. The coefficients of the equations as the eigenvalues and eigenvectors of the structural modes are developed in a finite element analysis (FEA) software tool, like I-DEAS, and are then imported into MATLAB.

The optical model, the other dynamic virtual telescope, is built in the optical design package which incorporates user-defined surfaces written in C and/or custom ray trace algorithms. The compiled C code is linked to the ray trace software, like OSLO or CODE V, as a dynamic link library (DLL) file. To synchronize the optical virtual telescope with the structural model, changes are made to the definition of the user-defined surfaces via a data-exchange interface.

A key element of the interface is a transfer data structure equally supported in both domains. In our current model this data structure comprises the rigid body displacement of each optical surface, i.e. all the segments and the secondary mirror. The rigid body displacements are actually calculated as the  $x$  and  $y$  de-centers and the first three Zernike coefficients of the surface (piston, tip and tilt). In the future it is possible to expand the data structure with shape deformation information by simply extending the number of Zernike terms considered.

The same algorithm is used in both packages to generate the telescope, and especially the segmented primary mirror model. Although it is exceedingly useful for the ray-tracing software to support external C language routines, this is not a serious restriction, as many commercial packages provide such support. Due to the use of the same algorithm, the coordinate systems for the two virtual telescopes are the same. However, this is not a firm requirement; the synchronization process involves several coordinate transformation steps anyway, because it is convenient to move the surfaces in a local coordinate system.

The second significant challenge an integrated telescope model presents is the propagation of several light beams through environments of vastly different character. Although the beam path of greatest final importance belongs to the science object (SO) in the sky, the proper operation of the telescope depends no less on beams from natural and possible laser guide stars (NGS and LGS). Light propagates through large distances in the atmosphere as a collimated (SO, NGS) or nearly collimated (LGS) beam. When the light reaches the telescope, it propagates through the optical systems associated with active optics, adaptive optics and astronomical instrumentation. During this process, the spatial scale of the light wave rapidly decreases before intersection with the image plane. In the atmospheric section of the beam the random perturbation of the light phase is the major disturbance. In the telescope and optical instrumentation, the dominant effects are geometric changes.

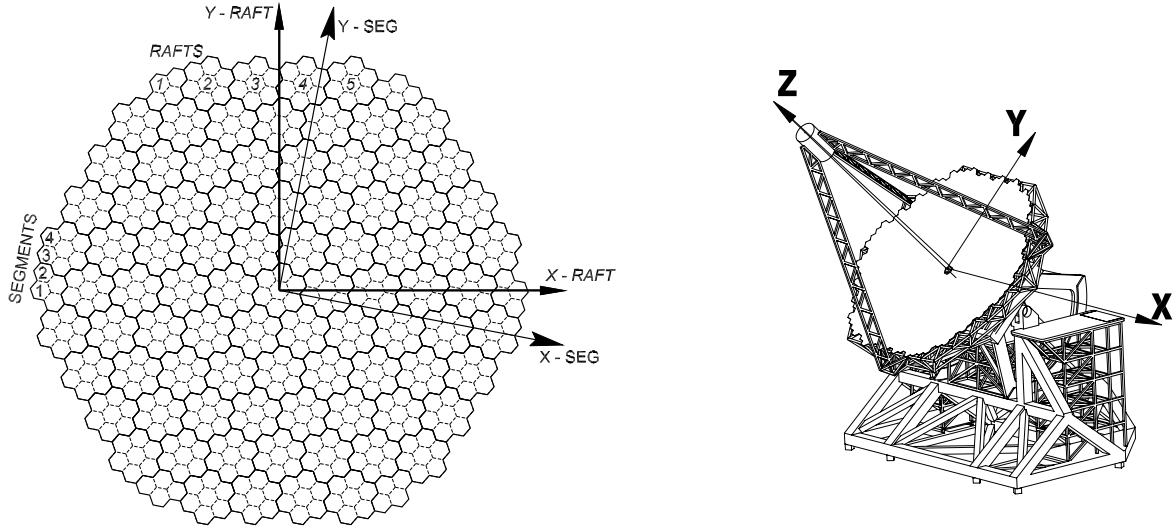
It is convenient to use different techniques to describe these beam in the different regimes: the atmospheric part of the beam is usually simulated by Fourier propagation; the interaction of the beam with the optical surfaces in the telescope and instruments is easily handled with ray tracing.

A technique interfacing wave propagation and ray-tracing domains is crucial in enabling an integrated telescope model built on commercial tools. We use the optical path difference (OPD) at the pupil positions and the geometry of the given pupil as the intermediate data structure to transfer data between the different propagation domains. The OPD is expressed in length, rather than in wavelengths or phase, to avoid wavelength dependence. The assumptions allowing the use of the OPD are: (i) the amplitude distribution across the interface pupil is fairly uniform, (ii) the refraction index as a function of wavelength is reasonably constant in the optical band used.

The advantages of using the OPD as the transfer data structure include its close relationship to conventional measures of imaging performance, like Strehl ratio and point spread function (PSF), its quasi-linear dependence on both deformation and refractive index, and its independence of wavelength. The details of the light domain interface are explained in Section 4.

### 3. COMPUTATIONAL ENVIRONMENT

The current model of a thirty meter telescope integrates three different software packages: I-DEAS for structural analysis (FEA), MATLAB/Simulink for control simulation, and CODE V<sup>9</sup> for ray tracing and optical measures. In each of these software tools a virtual telescope is created. In this telescope there are rafts that support seven hexagonal segments each, except for six rafts on the corners of the primary which hold four segments each, and the center raft which holds six segments. To communicate between the software tools, a common labelling of the segments and the rafts must be agreed upon and enforced. Figure 2 is the labelling contract that each telescope model must adhere to.



**Figure 2.** The schematic of the primary mirror on the left illustrates the numbering system used within the integrated model to identify the segments and the underlying raft support structure. The telescope on the right defines the 3-D telescope coordinate system.

The FEA needs to be run only once, before the simulation of the telescope begins to run in MATLAB/Simulink. Therefore, the FEA is considered to be independent of the computational environment for the simulation. The output of the FEA is a modal system of differential equations that is read into MATLAB/Simulink using IMAT.<sup>10</sup> Then, a state space model of the deformations is constructed in MATLAB, which is solved numerically to give the deformations on both the primary and secondary mirrors.

The deformations to the mirrors are imported into the virtual telescope in the optical model. A ground layer atmospheric phase screen is also imported into the virtual optical telescope. The optical model returns the OPD of the light passing through the telescope model. The model proceeds in four main steps:

- Calculate the structural deformations of the telescope based on input forces. The environmental perturbation forces are precalculated,
- Transfer the structural deformations to the optical kernel,
- Calculate the OPD based on the new deformation data and the precalculated phase screens.
- Pass the OPD back to the control simulation.

Figure 3 shows the links of the different software packages. Facilitating the communication between the control simulation and the optical kernel is one of the main challenges of the computational environment.

### 3.1. Serial model on Windows

The first implementation of the integrated model was realized as a serial model that runs on one machine in a Microsoft Windows environment. This environment is common to all of the desktop machines in the integrated modeling team. Also, most of the commercially available optical tools can only run in a Windows environment.

Several optical packages are available to act as an optical server to the control simulation. Almost all of these use Microsoft Windows technologies such as DDE (Dynamic Data Exchange) or COM (Component Object Model). For this version of the integrated model, CODE V was chosen to act as the optical kernel. CODE V v9.30, has a wide variety of services that are available through its COM interface. When the simulation begins

in MATLAB, a connection is created that links MATLAB to CODE V. After this connection is created, commands are sent to CODE V that load the optical kernel representation of the telescope. Initially the telescope in both the optical kernel, and the control simulation are unperturbed.

The control simulation in MATLAB/Simulink is ready to begin. The system of differential equations defined by the state space system of equations is solved using a variable time step solver. Forces are applied to the virtual structural telescope. The deformations of the virtual structural telescope are translated to tip/tilt and piston for every segment and sent to CODE V via the COM connection.

The displacement of the secondary mirror is also sent, this time using five different degrees of freedom:  $x$  de-center,  $y$  de-center,  $z$  de-center,  $x$  rotation, and  $y$  rotation. Commands are again sent to the optical kernel and a request is made for an Optical Path Difference (OPD) map based on the deformations applied to the virtual optical telescope in the optical kernel. Currently the optical kernel is returning a grid with  $511 \times 511$  elements, and the data transfer of this many data points may slow down a simulation. As a result, the request for an OPD is only made at fixed time intervals, while the deformations to the virtual structural telescope are updated at every variable time step of the differential equation solver.

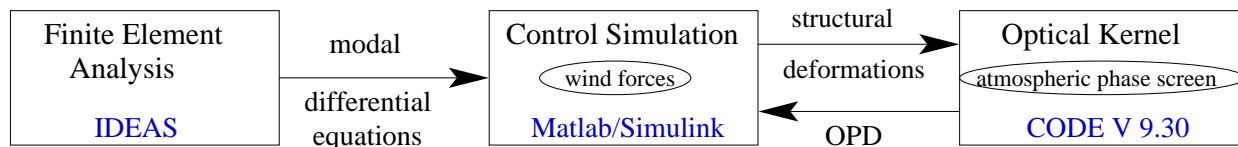
The simulation in MATLAB/Simulink may continue running for as long as there are structural forces, or until the user requests the simulation to stop. At the end of the simulation it is very important that CODE V is told to stop, and then the COM object from within MATLAB is destroyed. Having the Windows task manager open is a useful tool during development of this integrated model.

### 3.2. Serial model on Linux

As the complexity of the model increases, so does the time taken to run the model. To achieve faster run times, the integrated modeling team would like to split up the work to be done and run the computation on a cluster of workstations running Linux. Such a cluster is commonly referred to as a *beowulf* cluster. The first step on the way to running the developed model on a cluster of Linux servers, is running it on one Linux server, or having a serial model on a Linux machine. A change in operating system venues requires another look at all of the pieces of the integrated model.

Since the FEA is a preprocessing step, it can be run any system that supports the FEA software, and then imported to MATLAB on Linux for the control simulation. The control simulation running in MATLAB also does not pose any kind of problem since MATLAB is available for several different operating systems, including Linux.

Difficulties arise when considering the optical kernel. The optical kernel chosen for the first version of the integrated model of the large optical telescope is CODE V, and CODE V is not available for the Linux operating system. One possible solution is to have a computer separate from the Linux cluster running Windows, and have that machine handle all of the optical kernel calls. This may work if the simulation is running on one Linux server, but when the computation is divided to run faster, a bottle neck is created at the optical kernel component. The bottle neck can be alleviated by having a separate cluster of windows machines to handle the optical kernel component of the computation that was the same size as the Linux cluster. The Linux servers sit idle during optical kernel computation, and the Windows machines sit idle during the structural



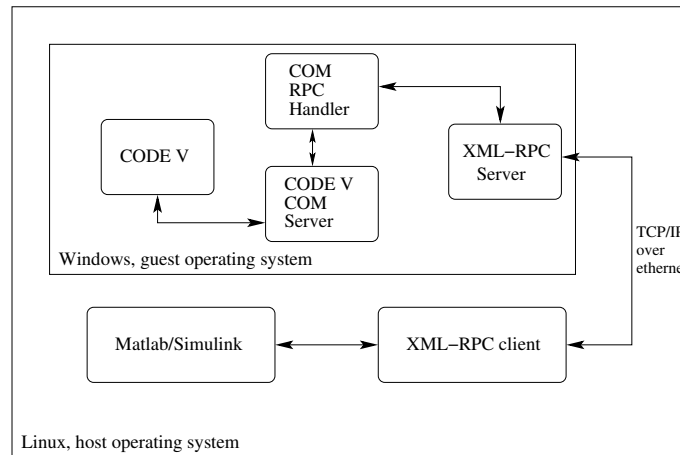
**Figure 3.** The arrows in this block diagram represent data being transferred from one software tool to another. The data passed between the FEA block and the Control block happens before the simulation begins. The data passed between the Control block and the Optical block happens at fixed intervals during the simulation. The circled values represent precalculated values used within the blocks.

control component of the computation. This is an inefficient use of resources, and requires twice the amount of hardware than running everything on the Linux operating system. Therefore, our solution is to run the entire model on the Linux operating system.

There are several ways to run a Windows program on the Linux operating system. There is a Windows emulator, called wine, that will trap all of the Windows operating system calls that a piece of software sends out, and translate those operating system calls to the appropriate Linux counterpart. Since Windows doesn't release the exact meaning of all of their operating system calls, and sometimes the Linux counterpart to such calls is ambiguous at best, running a Windows program under wine can be error prone. Also, the optical kernel is controlled by the control simulation, which requires a great deal of communication between the two components. Wine is still not developed enough to handle the needs of an integrated model on Linux.

The serial integrated model on Linux is realized with virtual machines. VMWare's flagship product, VMWare, can create a virtual machine inside of Linux. VMWare is a *hardware* emulator. After the installation of VMWare when you start the virtual machine it opens a virtual machine that has nothing loaded on it. Then, to run the optical kernel, Windows must be installed on this blank virtual machine; this is done the standard way with Windows installation disks. In the VMWare nomenclature Linux is called the "host" operating system and Windows is called the "guest" operating system. During the configuration of the virtual machine an IP address is chosen for the guest operating system. This IP address is used for communicating across the virtual machine layer.

With the control simulation running on the host operating system and the optical kernel running on the guest operating system, the COM protocol is no longer directly usable from MATLAB. Instead, the deformation data is sent to a communication server on Windows, which in turn sends the deformation data to CODE V with the COM standard. On the return trip, the OPD map is sent from CODE V to the communication server on Windows, which sends it back to MATLAB on Linux. The communication protocol used to send the information across the guest/host operating system boundary is XML-RPC. There are several languages that implement the XML-RPC communication standard, and this model is using Perl.<sup>11</sup> The XML-RPC client includes the package `Frontier::Client`<sup>12</sup> and the XML-RPC server uses the packages `Frontier::Server`<sup>12</sup> and `Win32::OLE`<sup>13</sup> to communicate with CODE V. Figure 4 shows a diagram of all the communication channels that the data flows through.



**Figure 4.** Structural deformation data is created in the model of the telescope in MATLAB/Simulink on a Linux server. The data follows the arrows and are then applied to the optical model of the telescope in CODE V running on the guest operating system, Windows. After the deformations are applied the OPD is returned to MATLAB.

### 3.3. Parallel model on Linux

The foundation to create a parallel model on a Linux cluster is established. There is a serial model on Linux, and now the computation needs to be split and run in parallel. The communication speed disadvantage in the Linux serial model should be recovered once the computation is parallelized. This is the next step for the integrated modeling team.

## 4. OPTICAL MODEL

In this section, the optical software requirements to model GSMT, and the optical properties of GSMT are discussed. In addition, a novel technique using sequential ray-tracing to propagate light in a segmented mirror telescope is presented.

The GSMT is nominally a Ritchey-Chretien telescope with the optical properties shown in Table 1.

**Table 1.** First and third order properties of the GSMT. The dimensions are in units of meters and the stop is located at the secondary mirror.

	ROC (m)	Z Position (m)	Semi-Diameter (m)	Conic Constant
<b>Primary</b>	60	0	16	-1.0004
<b>Secondary</b>	4.2254	28	1	-1.2381

The primary mirror is filled with 618 hexagonal segments that are 1.33 m in diameter from straight edge-to-straight edge. The segments are hexagons when projected onto the  $xy$  plane. The hexagonal segments are nominally arranged in groups of seven. Each raft of seven is supported by a single mechanical support module. There are a total of 91 rafts. Figure 2 is a schematic of the primary. The raft axes are separated by  $-10.89$  degrees from those of the segments. The coordinate system of the telescope is aligned with the rafts.

The concept of the integrated model is to use commercially available software packages that are dynamically linked with MATLAB as “modules” of the integrated model. In this context it is determined that the optical software package used to evaluate the optical performance of the telescope should contain the following capabilities,

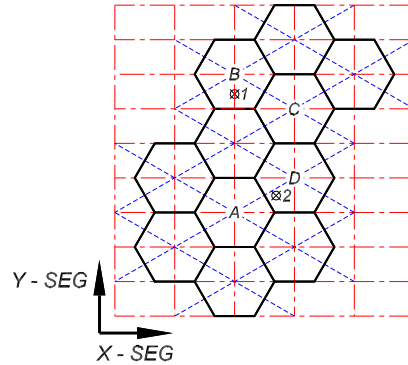
- Dynamic inter-software communication link with MATLAB client.
- Efficient evaluation of telescope optical performance using ray tracing.
- Graceful management of the irregular primary boundary.
- Fast ray trace algorithm with minimal overhead.
- User-defined surfaces and custom ray traces that can be constructed from a familiar high level programming language.

### 4.1. Segmented primary mirror model with sequential ray tracing

The segmented primary mirror is a user defined surface written in C and linked to the ray trace software via a DLL. The algorithm generating the user defined surface sets up a rectangular grid with the segment centers falling on the grid points as seen in Figure 5. The grid size is 0.665m and 0.575m along the  $x$  and  $y$  directions, respectively. Note that only half of the grid points are segment centers. A similar grid, rotated and scaled, is defined for the raft centers.

The generating algorithm places segments at the potential grid points subject to the requirements that they be: (i) within a specified radius of the mirror center; (ii) not sole members of a supporting raft; and (iii) not inside the central hole. The gap between segments can be modeled by placing a somewhat smaller segment at a given grid point.

An ordinary (sequential) ray tracing algorithm is used to trace rays through the optical model. Non-sequential algorithms for handling the multiple segments are likely to be slower and are unnecessary because the geometry of segments encountered is known *a-priori*.\*



**Figure 5.** A schematic of the algorithm in the DLL used to define the boundaries of each hexagonal segment. An array of rectangles is defined on the centers of the array of hexagons. We note that the skew edges of the hexagons form perpendicular bisectors of the lines joining opposite corners of the rectangles. This suggests a simple criterion for identifying the hexagon intercepted by a ray.

As shown in Figure 5, the same rectangular grid defining the segments is used to determine the which segments are struck by the rays during the ray trace. The steps in the ray-tracing algorithm are as follows:

- Use the same grid on the primary mirror that was used to generate the mirror. Each rectangle formed by this grid is identified by the (integer) coordinates of the lower left corner.
- Find the rectangle that contains the ray intersection point by dividing the  $x,y$  ray coordinates by the  $x,y$  rectangle pitches. The integer parts (floor) of the quotients are the  $x,y$  indices of the rectangle in the array. There are only two hexagons per rectangle, so the number of possible segments that the ray intersects is reduced from 618 to 2. Note that only half of the rectangle grid points correspond to segment centers.
- Determine which of the two hexagons the ray strikes: Once the rectangle indices are calculated, a test is performed to see which of the two hexagons is intersected. For rectangles whose lower left corner is a hexagon center, a line from the bottom left corner to the top right corner of the rectangle is drawn. See the example of the rectangle whose lower left corner is the point *A* in Figure 5. For the other half of the rectangles, a line is drawn from the top left corner to the bottom right corner of the rectangle. The hexagonal boundaries of neighboring segments bisect the lines equally. The hexagon intersected is that whose center is closest to the ray intersection.
- Once the hexagon indices is determined, a table of valid segment centers is consulted to determine whether the ray fell on a glass segment or missed the primary. (In this manner the irregular boundary is computed as well.)

The ray trace is performed using the parent asphere associated with the segmented mirror. The parent mirror is a full sized monolith in coincidence with the segment at the location of the segment.

---

\*It should be noted that the all the optical software used by NIO provides non-sequential ray-tracing capabilities. Different packages are used for diffraction, stray light and general optical analysis.

## 4.2. Introducing geometric perturbations

Once the segments the rays strike are determined, perturbation data from the virtual structural telescope are passed to the virtual optical telescope, in the form of  $z$  de-center (piston),  $x$  tilt, and  $y$  tilt. The rays are then traced in their reflection from the segments. The tilt perturbations are introduced to each segment as a tilt of the parent mirror about its global origin. Tilting about the global origin also results in an inconvenient translation of the segment, as shown in Equation (1).

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = R_{XY} \begin{pmatrix} x_{center} \\ y_{center} \\ 0 \end{pmatrix}. \quad (1)$$

The rotation matrix is calculated by applying the matrix product of the rotation matrices associated with  $x$ ,  $y$ , and  $z$  tilt sequentially. In order to counteract the translation it is subtracted from the rotated segment position vector. The resulting position vector  $(x', y', z')$  of a given ray due to the perturbation tilts is,

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R_{XY} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} - \begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \Delta y \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ \Delta z \end{pmatrix}. \quad (2)$$

The position vector  $(x', y', z')$  is in the same frame-of-reference as the rotated parent. Therefore, the sag function that describes the shape of the parent conic is the simple sag function for a general conic. This is the case for every ray traced. Hence, the shape of the surface is defined in its explicit form,

$$f(x', y', z) - z = \frac{(x'^2 + y'^2)}{R_B \left[ 1 + \sqrt{1 - (1 + \kappa)(x'^2 + y'^2)/(R_B^2)} \right]}, \quad (3)$$

where  $R_B$  is the base radius of curvature of the conic. The partial differentials of the surface are also calculated. The partial differentials are used by the ray trace kernel to iterate the ray until it encounters the surface.

Perturbations to the secondary mirror are modeled as rigid body movements. Hence, coordinate breaks are inserted before and after the secondary in such a way that tilts and de-centers to the secondary are counteracted to preserve the coordinate system of the virtual optical telescope.

## 4.3. Introducing atmospheric phase perturbations

An important function of the integrated model is to include the effects of atmospheric turbulence. To accomplish this an array of *Kolmogorov*<sup>14</sup> phase screen data is passed to the ray trace model. The current implementation of the optical module only allows phase screen data to be passed to the virtual optical telescope by file I/O. In CODE V v9.30, this data is passed as an interferometer (INT) ASCII file. The INT file contains a header with information necessary to convert real optical path differences into 16 bit integers. Each frame of the phase screen is read into CODE V sequentially. The results of introducing a ground layer phase screen into the entrance pupil are discussed in Section 6.

Phase screens from multiple-layers can be introduced into the optical model. This is achieved by propagating the phase screens from the given altitude using a Fresnel diffraction Kernel.<sup>15</sup> The phase screens are typically 60-100 m in extent and are propagated distances on the order of 20-30 km. For a wavelength of 1  $\mu m$ , the propagation has Fresnel numbers on the order of  $4.10^4$ . For such high Fresnel numbers, the dominant effects of turbulence several kilometers above the telescope are phase perturbations at the telescope entrance pupil and scintillation can be neglected.

## 5. STRUCTURAL MODEL

It is customary to introduce three different description tools for a mechanical structure. They represent increasing levels of abstraction. From the integrated modeling standpoint their relationship is what is important, i.e., how the real world can be approximated from the abstract mathematical model.

The nodal representation can be considered as the physical, or real system, although it already has some level of approximation and abstraction through the choice of nodes. The associated multidimensional coordinate system  $\mathbf{q}$  is defined through the six dimensional physical displacements of nodes (the  $x$ ,  $y$ , and  $z$  physical displacements and the rotations about the  $x$ ,  $y$ , and  $z$  axes). A nodal model of a structure is characterized by the mass ( $\mathbf{M}$ ), damping ( $\mathbf{D}$ ), and the stiffness ( $\mathbf{K}$ ) matrices, as well as the input ( $\mathbf{B}_0$ ) and output ( $\mathbf{C}_0$ ) operators (mask matrices).

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} &= \mathbf{B}_0\mathbf{u} \\ \mathbf{y} &= \mathbf{C}_0\mathbf{q} \end{aligned} \quad (4)$$

The real degrees of freedom of the system ( $\mathbf{q}$ ), in general, are not orthogonal. That means the differential equation system of Equation (4) is highly coupled. In contrast, the modal coordinate system is defined through the strengths of structural modes ( $\mathbf{q}_m$ ). Each individual node displacement is the linear combination of these modes,  $\mathbf{q} = \Phi\mathbf{q}_m$ , where  $\Phi$  contains the mode shapes as columns. The enormous advantage of modal representation is that since the modes constitute an orthonormal basis, it de-couples the system of differential equations.

$$\begin{aligned} \ddot{\mathbf{q}}_m + 2\mathbf{Z}\Omega\dot{\mathbf{q}}_m + \Omega^2\mathbf{q}_m &= \mathbf{M}_m^{-1}\Phi^T\mathbf{B}_0\mathbf{u} \\ \mathbf{y} &= \mathbf{C}_0\Phi\mathbf{q}_m \end{aligned} \quad (5)$$

A finite element analysis tool, like I-DEAS or NASTRAN, can provide the coefficient matrices of Equation (5), since  $\mathbf{M}_m = \text{diag}(m_i)$  is the diagonal matrix of modal masses,  $\Omega = \text{diag}(\omega_i)$  is the diagonal matrix of resonant frequencies, and  $\mathbf{Z} = \text{diag}(\zeta_i)$  contains the modal damping coefficients.

The modal state representation reduces the order of differential equations describing the system by defining a state vector as the concatenation of modal displacements and velocities,  $\mathbf{x} = [\mathbf{q}_m^T \dot{\mathbf{q}}_m^T]^T$ . This formalism expresses the basic system characteristics, the inputs, the outputs, and the dynamics of the system in a quite compressed and visual way.

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\Omega & -2\mathbf{Z}\Omega \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}_m^{-1} \end{bmatrix} (\Phi^T\mathbf{B}^{<1>}\mathbf{u}^{<1>} + \Phi^T\mathbf{B}^{<2>}\mathbf{u}^{<2>} + \dots) \\ \mathbf{y}^{<1>} &= \mathbf{C}^{<1>}\Phi \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} \\ \mathbf{y}^{<2>} &= \mathbf{C}^{<2>}\Phi \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{x} \end{aligned} \quad (6)$$

Equation (6) clearly shows a core dynamic system with matrix operators facilitating different input/output groups. The bracketed superscript numbers refer to input and output structural node groups, like primary (<1>) and secondary (<2>). Although the input and output operators appear to be the result of matrix multiplication ( $\Phi^T\mathbf{B}^{<1>}$  or  $\mathbf{C}^{<1>}\Phi$ ), they are actually truncated versions of the mode shape matrix  $\Phi$ . For each input or output, only the rows corresponding to the coordinates (DoF) participating in the given input or output are kept. The rest of the rows are deleted. So, by identifying DoF groups that can act as input and/or output conduits, Equation (6) can be generalized.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}^{<core>}\mathbf{x} + \mathbf{B}^{<core>}(\tilde{\Phi}^{<1>T}\mathbf{u}^{<1>} + \tilde{\Phi}^{<2>T}\mathbf{u}^{<2>} + \tilde{\Phi}^{<sup>T}\mathbf{u}^{<sup>}) \\ \mathbf{y}^{<1>} &= \tilde{\Phi}^{<1>}\mathbf{C}^{<core>}\mathbf{x} \\ \mathbf{y}^{<2>} &= \tilde{\Phi}^{<2>}\mathbf{C}^{<core>}\mathbf{x} \end{aligned} \quad (7)$$

Since the structural modes constitute an orthonormal basis of the possible structural deformations, the number of modes equals to the number of degrees of freedom in the system. Typically, this number is quite high, which

makes the integration of Equation (7) tedious. However, an overwhelming majority of the modes have little or no influence on the optical performance of the telescope, and there are well-developed methods to find and keep only the significant ones.<sup>16</sup>

Large segmented mirror telescopes pose a unique challenge for modal reduction, as the large number of segments implies an even larger number of optically significant degrees of freedom. In order to adequately characterize the optical behavior of the primary mirror, a large number of high spatial order modes must be kept in the model. Besides the obvious problems of handling thousands of modes, the high spatial order modes have high associated resonance frequencies, which in turn requires very small integration time steps and slows down the simulation.

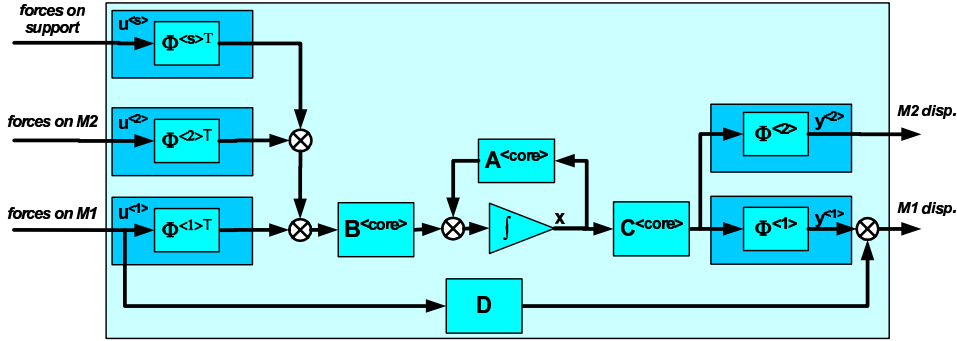


Figure 6. Outline of the structural model.

Fortunately, the limited temporal bandwidth of the mechanical excitations (wind forces, gravitational and thermal disturbances) provide a way to further reduce the number of dynamic modes included in the structural model. In our current model, all the high frequency modes necessary to describe the segment rigid body motions, but dynamically not excited, are set static. In other words, the segments are considered in the finite element model of the support truss as lumped masses, but those masses are neglected in the dynamics of the individual segments. The segments are connected to the support system by means of three actuators with stiffness of 10 N/ $\mu$ m, facilitating the tip/tilt and piston rigid body motion of the segments.

The segment displacements corresponding to actuator deformations due to forces on the primary mirror are considered as feed through ( $Du$ ) in the state equations (Figure 6, Equation 8).

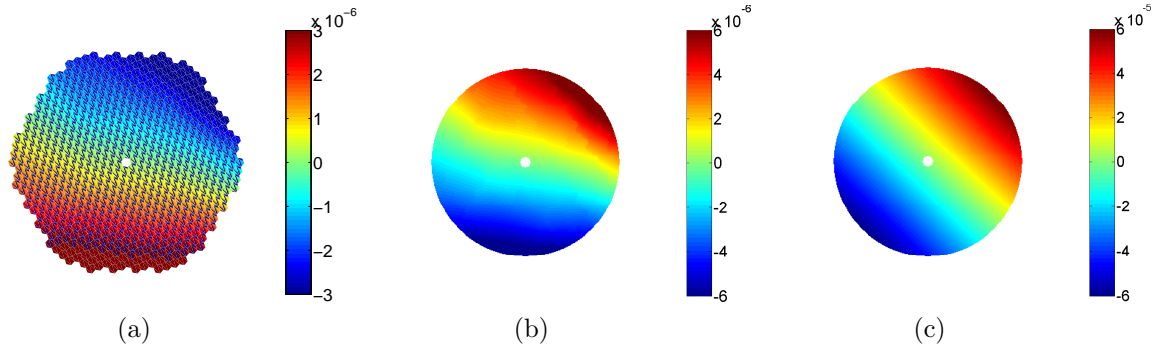
$$\begin{aligned}
 \dot{x} &= A^{<core>}x + B^{<core>}(\tilde{\Phi}^{<1>T}u^{<1>} + \tilde{\Phi}^{<2>T}u^{<2>} + \tilde{\Phi}^{<sup>T}u^{<sup>}) \\
 y^{<1>} &= \tilde{\Phi}^{<1>}C^{<core>}x + Du^{<1>} \\
 y^{<2>} &= \tilde{\Phi}^{<2>}C^{<core>}x
 \end{aligned} \tag{8}$$

## 6. PRELIMINARY RESULTS

### 6.1. Wind forces on the structure

The structural deformations used for the validation of the integrated model were derived from realistic conditions. The structural model of the telescope was subjected to dynamic wind forces acting on the secondary mirror. The time-dependent vector wind velocity applied is a sample of real wind velocity data obtained in measurements around the secondary mirror of the Southern Gemini telescope.<sup>17</sup> The mean wind speed was about 5 m/s. The drag forces were calculated by assuming a cylinder of 2 meters diameter and height. The primary mirror and the secondary mirror support legs were not exposed to wind to avoid the higher order deformations, which are well corrected by the edge sensor feedback loops.<sup>18</sup>

In both the optical and structural virtual telescopes, the positive  $z$  axis points from the primary to the secondary (Figure 2). As a result a negative deformation in the primary mirror creates a longer optical path through the system. Also, since the ray must travel to the segment, and then bounce back again to the secondary mirror, the OPD should be approximately twice the structural deformation. Figure 7 illustrates the results.

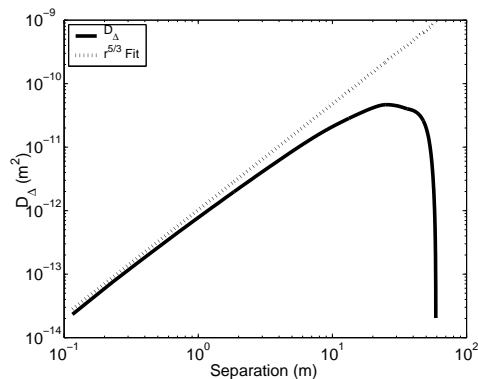


**Figure 7.** Figure (a) is a snapshot of the primary mirror. Figures (b) and (c) are the corresponding OPD snapshots without and with secondary mirror movement considered, respectively. All units are in meters.

## 6.2. Incorporating atmosphere

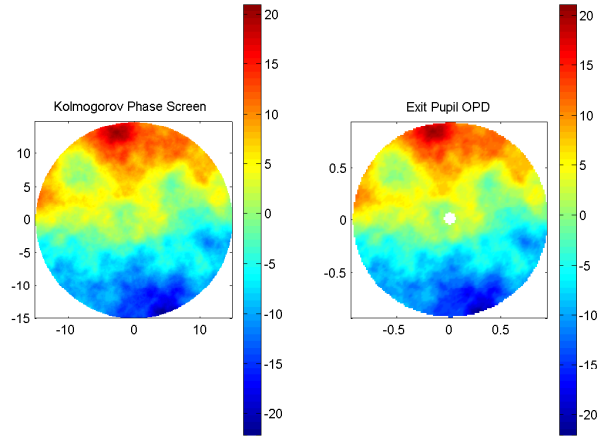
A ground layer Kolmogorov phase screen is generated using MATLAB. The parameters for the phase screen are, a Fried diameter ( $r_0$ ) of 15 cm, an infinite outer-scale, wavelength of  $0.5 \mu\text{m}$ , and a spatial sampling of 12 cm. The sampling rate of the optical system is set to 20 Hz and the wind velocity is set at 5 m/s along the  $x$  direction. Therefore the spatial increment of the phase screen between samples is 0.25 m. The Taylor frozen flow hypothesis is assumed.<sup>14</sup> To shift the phase screen along  $x$ , Fourier interpolation is used.<sup>15</sup>

Figure 8 is a plot of the phase structure function for the Kolmogorov phase screen introduced. The phase structure function has the predicted  $5/3$  power law behavior when plotted in log log space. Since the size of the phase screen is finite (60m), the structure function deviates from the power law at separations close to the screen size.



**Figure 8.** The phase structure function of the phase screen.

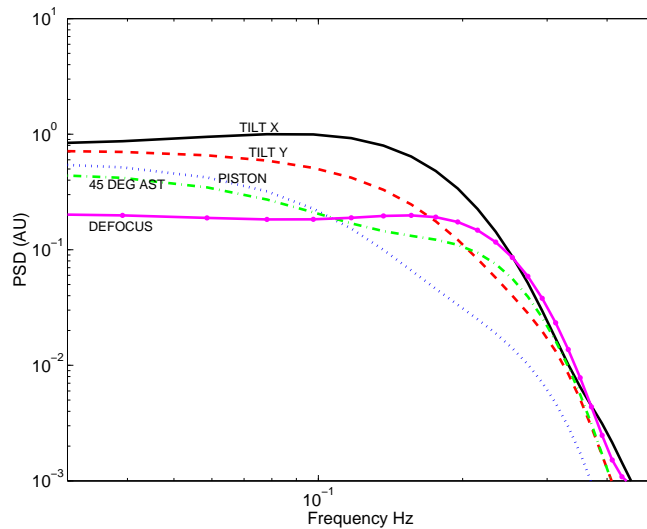
Figure 9 shows a mesh plot of the atmospheric phase screen before and after being introduced to the optical model. As can be seen the phase screen after being introduced into the optical model, which is evaluated in



**Figure 9.** The phase screen before and after being introduced to the optical model. The phase screen in the optical model is evaluated in the exit pupil.

the exit pupil is identical to the phase screen at the entrance pupil apart from the pupil magnification and the removal of piston error.

Figure 10 contains plots of the power spectral densities (PSDs) of the first five fringe Zernikes for the Kolmogorov screen. The Zernike data are obtained from CODE V Zernike fitting of the OPD maps in the GSMT exit pupil. The PSDs confirm that most of the power in the Kolmogorov screen is contained in the



**Figure 10.** The power spectral densities of the first five fringe Zernikes evaluated using CODE V.

lowest orders, which are the tilt Zernikes. The power of the tilt Zernike along the  $x$  direction is greater because the screen is shifting the  $x$  direction. The PSDs have a characteristic bandwidth due an apparent outer scale, which occurs at 0.16 Hz. The corresponding spatial frequency is calculated from the wind velocity. The apparent outer scale is the telescope entrance pupil diameter.

## 7. CONCLUSIONS

Our team is in the process of establishing and improving a tool set for ground based telescope modeling. We do not aim for a uniform software package built from scratch for this goal, but rather for an environment allowing the use of several general packages in concert. At this stage of the development we can report a number of noteworthy achievements:

- An algorithm that links the virtual structural and virtual optical telescopes residing in different software packages has been developed.
- An algorithm that generates a description of highly segmented primary mirrors in different programming environments (MATLAB and C) has been developed.
- A custom ray-trace algorithm attached to optical packages capable of accepting C routines has been developed.
- A software environment that allows different operating systems to work together and with communication between them has been developed, which opens up the road to parallelization of the model.
- The interface between Fourier propagated optical fields and ray-traced beams has been tested.

The major advantages of our approach are:

- The huge user community of the leading commercial software tools integrated into the model improves the credibility of the results. It also helps in developing components of the model, because of: (i) the well-known character of limitations and workarounds of the tools; (ii) the well established customer support and training; and (iii) the sizeable expert base available, and
- The reduced need for model transformation into the high-level simulation environment eliminates extra effort of the project team, and helps to keep the system model in synchrony with the actual detailed design.
- Use of commercial software reduces the cost and development time of the model software.

The price we have to pay for these advantages is the software complexity of the modeling environment; however, integrated models of sophisticated systems, like a Thirty Meter Telescope, tend to grow and become so computationally intensive that they require significant software support even in a uniform environment.

## ACKNOWLEDGEMENTS

The authors wish to acknowledge Brent Ellerbroek for his comments and suggestions regarding the *Kolmogorov* phase screen and Fourier beam propagation.

The New Initiatives Office is a partnership between two divisions of the Association of Universities for Research in Astronomy (AURA), Inc.: the National Optical Astronomy Observatory (NOAO) and the Gemini Observatory. NOAO is operated by AURA under cooperative agreement with the National Science Foundation (NSF). The Gemini Observatory is operated by AURA under a cooperative agreement with the NSF on behalf of the Gemini partnership: the National Science Foundation (United States), the Particle Physics and Astronomy Research Council (United Kingdom), the National Research Council (Canada), CONICYT (Chile), the Australian Research Council (Australia), CNPq (Brazil) and CONICET (Argentina).

## REFERENCES

1. L. Stepp, L. Daggert, and P. Gillett, "Estimating the cost of extremely large telescopes," *Proceedings of the SPIE* **4840**, pp. 309–321, 2002.
2. *Enabling a Giant Segmented Mirror Telescope for the Astronomical Community*, <http://www.aunio.edu/book/index.html>, AURA New Initiatives Office, 2002.
3. M. Lieber, "Development of the Ball integrated telescope model (ITM)," *Proceedings of the SPIE* **4757**, pp. 19–30, 2002.
4. R. Wilhelm *et al.*, "Integrated modeling for stellar interferometry - motivation, development strategy and practical usage," *Proceedings of the SPIE* **4757**, pp. 31–40, 2002.
5. D. Miller, O. de Weck, and G. Mosier, "Framework for multidisciplinary integrated modeling and analysis of space telescopes," *Proceedings of the SPIE* **4757**, pp. 1–18, 2002.
6. *MACOS Manual (Modeling and Analysis for Controlled Optical Systems)*, NASA Jet Propulsion Laboratory, 1999.
7. *Integrated Modeling of Optical Systems (IMOS), User's Manual*, NASA Jet Propulsion Laboratory, 2000.
8. P. Wallace, "A rigorous algorithm for telescope pointing," *Proceedings of the SPIE* **4848**, pp. 125–136, 2002.
9. *Optical Research Associates, CODE V Version 9.30 Reference Manual*, 3280 E.Foothill Blvd., Pasadena, CA 91107, 2003.
10. *IMAT - I-DEAS MATLAB Toolkit*, <http://ftp.ata-engineering.com/home.html?imat/bodySet.htm>, ATA Engineering, Inc., 2003.
11. L. Wall, T. Christiansen, and J. Orwant, *Programming Perl*, O'Reilly, 3 ed., 2000.
12. S. St-Laurent, J. Johnston, and E. Dumbill, *Programming Web Service with XML-RPC*, O'Reilly, 1 ed., 2001.
13. D. Roth, *Win32 Perl Programming*, New Riders, 2 ed., 2002.
14. J. Goodman, *Statistical Optics*, Wiley, 2000.
15. B. Ellerbroek and G. Cochran, "A wave optics propagation for multi-conjugate adaptive optics," *Proceedings of the SPIE* **4494**, pp. 104–120, 2002.
16. W. K. Gawronski, *Dynamics and Control of Structures; A Modal Approach*, Springer-Verlag, 1998.
17. M. Cho, L. Stepp, and S. Kim, "Wind buffeting effects on the Gemini 8m primary mirrors," *Proceedings of the SPIE* **4444**, pp. 302–314, 2001.
18. D. G. MacMartin and G. Channan, "Control of the California Extremely Large Telescope primary mirror," *Proceedings of the SPIE* **4840**, pp. 96–90, 2002.