Simulations of adaptive optics systems on 30-m-class telescopes

Matthew C. Britton, Richard G. Dekany

Matthew C. Britton, Richard G. Dekany, "Simulations of adaptive optics systems on 30-m-class telescopes," Proc. SPIE 4757, Integrated Modeling of Telescopes, (30 July 2002); doi: 10.1117/12.489822

SPIE.

Event: Workshop on Integrated Modeling of Telescopes, 2002, Lund, Sweden

Simulations of Adaptive Optics Systems on 30m Class Telescopes

Matthew C. Britton and Richard Dekany

California Institute of Technology

ABSTRACT

In this paper we describe the development of a C++ class library for the simulation of adaptive optics systems. This library includes functionality to simulate the propagation of electromagnetic waves through a randomly generated turbulent atmosphere and through an adaptive optical system. It includes support for extended emitters and laser guide stars, and for different types of wavefront sensors and reconstructors. The library also aims to support parallelization of simulations across symmetric multiprocessor and cluster supercomputers.

1. INTRODUCTION

Electromagnetic waves propagating to ground based telescopes are distorted by fluctuations in the index of refraction of air that arise in the turbulent atmosphere. These fluctuations introduce path length variations across the surface of a wavefront propagating through the atmosphere. The resulting relative phase fluctuations across the wavefront cause the image of an unresolved source formed by the telescope to depart from its ideal diffraction-limited form and acquire instead a broadened, distorted appearance. As air blows past the telescope, subsequent wavefronts acquire different sets of phase fluctuations, leading to a dynamic evolution of the point spread function that results in an image that may be many times broader than the diffraction limit.

The field of adaptive optics (AO) aims to compensate for these phase fluctuations by reflecting the wavefront off of flexible mirrors that deform to compensate for the optical path differences in the atmosphere. A number of AO systems are currently in operation, and have been successful in allowing astronomical observations that would otherwise not have been possible. However these systems have significant limitations described in more detail below that dictate the construction of a new generation of AO instrumentation of significantly greater complexity. In this paper we describe the design of a C++ class library intended to permit simulations of these new AO systems.

2. A BRIEF DESCRIPTION OF ADAPTIVE OPTICS

In order to motivate the functional and computational requirements of adaptive optics simulation software, we briefly review the operational concepts of these systems. Index of refraction fluctuations arise from a turbulent cascade in the atmosphere that is thought to be described by Komolgorov's theory of homogeneous isotropic turbulence.[1] In this theory atmospheric fluctuations are injected at some length scale called the outer scale. Nonlinearities in the Navier-Stokes equations mix energy at this length scale into smaller scale fluctuations. This generates a steady-state cascade from long to short spatial wavelengths. The cascade terminates at an inner scale where fluctuations are dissipated as heat. In the inertial range between outer and inner scales, the fluctuation spectrum is described by a power law of the form $\Phi(\kappa) = .033C_n^2\kappa^{-11/3}$ where κ is the spatial frequency. The power law coefficient C_n^2 depends on the height in the atmosphere, decreasing roughly exponentially above the ground. The values of C_n^2 can change on timescales of hours.[2]

Electromagnetic wavefronts propagating through the turbulent atmosphere acquire differential phase fluctuations across their surface. There is a length scale over which the wavefront seen at the ground will have roughly constant phase. This length scale is called the Fried parameter r_0.

$$r_0^{-5/3} = .423k^2 \int dz C_n^2(z) \tag{1}$$

Typical values of r_0 at good astronomical sites range from 10 to 30 cm at .5 microns. Note that r_0 is proportional to $\lambda^{6/5}$, so that the coherent patch size is larger at lower frequencies. The wind blows the turbulence over the

Integrated Modeling of Telescopes, Torben Andersen, Editor,
SPIE Vol. 4757 (2002) © 2002 SPIE · 0277-786X/02/$15.00

telescope aperture, generating a new realization of the wavefront's phase fluctuations on a timescale of order r_0/v, where v is the characteristic wind speed. Assuming a typical wind speed of 10 m/s and an r_0 of order 10 cm, the characteristic timescale of order 10 ms.

For simulations of wave propagation in the turbulent atmosphere, it is convenient to consider the atmosphere as a series of thin screens that modify the phase of an incident wavefront. One can accomplish this by dividing the atmosphere into a series of slabs and then integrating C_n^2 over the slab. In order for this approximation to be valid the slab must be much thicker than the outer scale in order to eliminate correlations between slabs, but must be thin enough so that diffraction effects may be ignored. Computationally one may generate a phase screen with the correct statistics.[3] Then one may propagate the wavefront through free space in between the screens and modify the phase of the wavefront at the location of the screen. This is sometimes called the split-step algorithm.[4]

Adaptive optics systems aim to compensate for fluctuations in the wavefront phase introduced by the turbulent atmosphere by correcting for the phase fluctuations in the wavefront collected by the telescope. These corrections must be performed on timescales of order milliseconds and on length scales of order r_0. To accomplish this, AO systems employ a system to sense the wavefront phase errors and a system to correct these errors. Since measurements of an astronomical science target are required, wavefront sensing of a reference source is done at a different frequency and a dichroic is used to separate the sensed light from the science light. For most astronomical AO systems sensing is done in the visible and science is done in the infrared. Typically the reference source must be brighter than about 13 magnitudes in the visible, though this depends on the details of the wavefront sensing scheme. These reference sources are often called natural guide stars (NGS) because they are almost always nearby bright stellar objects.

There are a number of different types of wavefront sensors. Perhaps the simplest to visualize is the Shack-Hartmann (SH) sensor, though it should be emphasized that there are several other potentially better types of sensors. The SH sensor sends collimated light from the reference source through an array of square lenslets with apertures that, when projected back to the primary, have typical size r_0. These lenslets focus subsections of the wavefont onto a detector. A phase gradient in the incident wavefront is thus converted into a shift in the centroid of the focused light at the detector. Information contained in these shifts may then be inverted to obtain the phase properties of the incident wavefront, at a sampling interval of order r_0. The inverse of this phase may be applied to a wavefront corrector so as to compensate for the delays caused by the atmosphere. Typically this corrector is a deformable mirror, in which actuators with a spacing of order r_0 push against a thin flexible mirror in order to generate the desired correction. The deformable mirror is positioned before the dichroic so as to compensate both the science light and the sensed light. In this way, errors in the wavefront after compensation by the deformable mirror are seen by the wavefront sensor as small deviations from a nominally flat wavefront. This mode of operation is called closed loop, and turns out to be significantly easier than open loop operation in which one attempts to sense and correct the uncompensated phase.

The inversion step mentioned above employs a matrix that relates SH wavefront sensor measurements to deformable mirror actuator commands. This matrix is called the reconstructor, and in the simplest case its specification requires only a knowledge of the wavefront sensor and deformable mirror geometries. Proposals for more elaborate reconstructors exist that incorporate the noise statistics of the wavefront sensor's detector, the influence function of the actuators on the deformable mirror, and even the atmospheric phase statistics.

The above description applies to almost all astronomical AO systems in operation - so called classical AO (CAO) systems. There are two major shortcomings to this scheme. First, because the wavefront sensor measures the path-integrated phase errors to the reference source, these errors are only valid over an angular scale called the isoplanatic angle. In directions outside the isoplanatic angle the integrated phase distortions differ from those in the direction of the reference source. Typical values of the isoplanatic angle are less than 10 arcseconds in the visible band. Together with the fact that the AO system requires a relatively bright reference source, the actual sky coverage that these systems achieve is only a few percent. This seriously limits the science that CAO systems may perform.

In order to improve sky coverage, backscattered light from laser beacons fired up into the atmosphere may be employed to serve as the reference source for the AO system. These are known as laser guide stars (LGS).

Both sodium and Rayleigh lasers have been experimentally employed to this end, with the former exciting a transition in the sodium layer at 90 km and the latter generating Rayleigh backscattering in the atmosphere below 20 km. A difficulty with laser beacons is that they probe the atmospheric turbulence in a conical beam, with apex at the beacon location and base at the telescope aperture. In contrast radiation from an astronomical source encounters turbulence in a cylindrical section of the atmosphere. The error generated by this mismatch is called focal anisoplanatism, and gets worse as the size of the telescope aperture increases and with decreasing height of the beacon.

To compensate for this, one may use an array of laser beacons that are positioned so as to probe the entire cylindrical column encountered by radiation from the astronomical source. Because lasers must be propagated up through the atmosphere and back down to the telescope, it turns out that their absolute position is not determined and additional NGS references must be employed to ascertain their true location. However these NGS references can be considerably fainter than their LGS counterparts. This technique is known as tomographic reconstruction. In reconstructing the atmosphere in this way, one derives information about the properties of the turbulence as a function of height rather than simply the path-integrated turbulence sensed in CAO. One may take advantage of this information to enlarge the isoplanatic angle through the use of additional deformable mirrors. Specifically, by placing deformable mirrors at positions optically conjugate to different heights in the atmosphere and applying the inverse of the phase measured through tomographic reconstruction, one correctly compensates for the phase errors over a much broader angle. The combination of multiple laser or natural guide stars and multiple deformable mirrors is known as multiconjugate adaptive optics (MCAO).

3. SIMULATION QUESTIONS

There are a large number of issues in adaptive optics that have not yet been explored with real systems. In this section we comment on a few such topics.

A wide variety of reconstructors have been proposed for both CAO and MCAO. The specific form that these reconstructors take depends on the properties of the wavefront sensor and deformable mirror, and may also include considerations of the atmospheric phase statistics. Many of these reconstructors have only been tested or simulated in a narrow context, and complete AO simulations would enable more direct comparative tests for these reconstruction schemes. In MCAO, the properties of the reconstructor are also tied closely to the details of the guide star brightness and asterism, the atmospheric C_n^2 profile, and the deformable mirror conjugate heights. These reconstructors depend upon nearly all components of the problem, and thus require a full simulation to test their performance.

In MCAO, the C_n^2 profile is one of the inputs required to formulate the reconstructor. Since the C_n^2 profile evolves on the timescale of hours, this profile must be measured contemporaneously with the science observations. Because of this, it is important to understand the sensitivity of the AO correction to errors in this profile. Simulations can help to indicate the level of sensitivity.

Sodium lasers offer the advantage of producing high altitude reference beacons that reduce the level of focal anisoplanatism, while Rayleigh lasers yield relatively brighter, low altitude beacons for a fraction of the cost. An MCAO system that utilizes at least some Rayleigh beacons may present a significant cost savings over a system composed entirely of Sodium lasers. It is also not clear whether it is better to have a few bright beacons or many faint beacons, or how best to distribute these beacons on the sky. Finally, the spatially extended nature of the backscattered radiation from these beacons may require a pulsed laser scheme. Exploration of this laser parameter space in simulation could yield a better understanding of the laser beacon properties required to attain a given level of correction.

There are also scientific reasons for running an AO system in open loop. By controlling the wavefront so as to manipulate the intensity distribution of an image, one can attempt to create regions of very low intensity within the image in order to achieve very high dynamic range detections. This is called the dark hole technique, and may allow imaging and spectroscopy of very faint planetary companions around nearby stars. An open loop adaptive optics system has not yet been demonstrated in astronomy, and simulation is a sensible place to start examining this type of system.

In addition to the above scientific motivations for simulating AO systems, confidence in the accuracy of the simulations must be assured through verification. One can ensure accuracy of the phase screen spatial frequency statistics by generating many such screens and accumulating these statistics. Free space wave propagation may be verified through comparison to analytic results such as propagation through a square aperture[5] or a circular aperture near the focal point, where the field strength is approximated by Lommel functions.[6] There are a large number of analytic results that describe wavefront amplitude and phase statistics after propagation through a turbulent atmosphere[7] for both geometric and diffractive propagation. Finally, the ultimate test for this sort of simulation is to attempt to reproduce the results of an operational adaptive optics system.

4. COMPUTATIONAL REQUIREMENTS

The previous section suggests that AO simulation software could be useful in addressing a large set of issues. Because exploration of these problems will require many sets of simulations, it is important to assess the computational requirements for the simulation. The generation of most types of reconstructors require inversion of a matrix. For MCAO reconstructors this matrix can be tens of thousands of elements on a side, and can contain gigabytes of data. The inversion of such matrices is a problem of considerable computational difficulty. However, software for the parallelization of large matrix inversions are already available. Assuming the reconstructor has already been created, the computational load in running the simulation will be dominated by the fourier transforms required to perform the wave propagation. We now estimate the computational requirements for this part of the problem.

First, let us estimate the typical size of the array that will be required to represent the wavefront. It should be emphasized that this will be a free parameter in the software, but this estimate will serve as a benchmark for the size of the calculation we expect to perform. We require spatial samples on the wavefront to be small enough that the wavefront phase does not differ by more than about π radians between samples - otherwise the fidelity of the simulation would be seriously compromised. This sample size is of order r_0, though this quantity is a statistical average and deviations may be expected. This would indicate that we need a sample size several times smaller. At the same time, in the case of SH wavefront sensors we would like to have a number of samples across each lenslet in the square lenslet array. These lenslets are of order r_0, and again we would like several samples in this interval. Let us assume an r_0 of 20 cm at .5 microns. Assumimg an aperture size of 30 meters and samples of size $r_0/8$, we require 1200 samples across the aperture. Since fourier transforms are typically optimized for arrays that have a length equal to a power of two, we would expect to round the array dimensions up to 2048x2048. We can also expect aliasing in the fourier transforms to spoil the edges of the array,[8] so we can use these additional points as zero padding around the edges of the array during the transforms. Therefore, a plausible array size for typical AO simulations of 30 meter telescopes would be 2048x2048. It should be noted that propagation of the infrared radiation would require a factor of 2 or less resolution.

Let us next count the number of transforms in an AO simulation. Except for the final far-field propagation to the detector, all free space propagation will be in the near field. The near field propagator requires two fourier transforms, while the far field one requires one.[9] Additionally, because the wind shifts the atmospheric layer with respect to the wavefront we may need to align the pixels in the atmospheric layer with those in the wavefront. This alignment may be accomplished by the fourier shift theorem,[10] where a phase slope is added to the fourier transform of the atmospheric layer in order to shift the pixel boundaries. So for each time step we require four transforms to propagate the wavefront through each layer, two transforms per optical element, and one transform to reach the detector. Additionally, laser beacons require propagation up through the atmosphere and back down, where as astronomical targets require just downward propagation. Therefore, the number of fourier transforms required to simulate a single source at a single time step is

$$\begin{pmatrix} \#\text{ of FT's} \\ \text{per time step} \end{pmatrix} = 2 * \begin{pmatrix} \#\text{ of} \\ \text{optics} \end{pmatrix} + 1 + 4 * \begin{pmatrix} \#\text{ of atm.} \\ \text{layers} \end{pmatrix} * \begin{cases} 2 & \text{(LGS)} \\ 1 & \text{(NGS)} \end{cases}$$

For a CAO simulation we require propagation of radiation from an NGS and at least one science target. Assuming 3 atmospheric layers and 5 optics this requires 23 fourier transforms per source. To simulate one

second of data for both frequencies assuming time steps of 1 millisecond, we require 23000 transforms at each frequency. A 1.7 GHz Pentium 4 processor can accomplish a 2048x2048 transform in about 1 second, and a 1024x1024 transform in about .25 seconds. Thus we expect the transforms alone to take about 8 hours on this processor, yielding a simulation efficiency of about 3E4 times real time.

In contrast MCAO simulations require many more transforms because there are more sources whose wavefronts must be propagated through the atmosphere and because of the additional deformable mirrors. More atmospheric layers are also required, so as to present a realistic atmosphere for tomographic reconstruction. Assuming 10 atmospheric layers, 4 deformable mirrors, and 4 additional optics, we require 57 transforms for an astronomical source and 97 transforms for a laser beacon. Assuming an MCAO simulation with 4 NGS and 8 LGS, the transformations will take roughly 275 hours for one second of simulated data, or 8E5 times real time. It should be noted that these calculations incorporated the minimum number of sources to effect the correction of the deformable mirror. Simulations in which a map of the Strehl ratio as a function of field position is required would necessitate the propagation of wavefronts from many more science targets.

5. PARALLELZATION

Given the large parameter space that we expect to explore with this simulation, the simulations described above are too large to pursue on a single workstation. A solution to this problem lies in parallelizing these simulations to run on supercomputers.

Supercomputers fall into two architectural classes: symmetric multiprocessor (SMP) machines and clusters. SMP machines consist of multiple processors that share the same memory and can communicate with each other quickly through this shared memory. In contrast, clusters consist of isolated processors that communicate at a much slower rate via ethernet. There are also hybrid systems, in which SMP machines are connected to form clusters. A cluster made of dual Pentium systems would be an example of such a hybrid system. The faster processor interconnect speed of SMP's is an advantage for problems in which a large amount of interprocessor communication is to be expected. This is the case for performing computations like matrix inversion, because the processors must communicate a large number of intermediate computations in order to divide the load. In contrast, clusters of Pentium workstations are cheaper to build. Because of this, these clusters tend to be more powerful than their SMP counterparts.

In general it is not possible to productively use an arbitrarily large number of processors to work on a given problem. Interprocessor communication overhead and computational interdependencies may dicatate a situation in which only a small number of processors can work at once. Fortunately, the problem of simulating adaptive optics systems is well-suited to parallelization, because much of the wavefront propagation may be performed independently. To see that this is the case, first consider a classical AO simulation. The only part of the simulation that introduces a computational dependency is the fact that information from the wavefronts detected by the wavefront sensor is used to update the deformable mirror surface, which subsequently modifies the next wavefront. This dependency forces the simulation to compute the results at each time step before proceeding to the next one. However, for wavefront propagation to the surface of the first deformable mirror all time steps are independent.

In this vein, one might consider dividing the simulation into two stages. In the first stage, wavefronts could be propagated to the telescope aperture and saved to disk. In this stage, one might use as many processors as were available, giving each processor the task of performing the propagation of a wavefront from a single source at a single time step to the ground. This stage accounts for more than half the transforms in a classical AO simulation, and a significantly higher proportion in an MCAO simulation. For example, in the case of 4 natural guide stars and 8 laser guide stars discussed above, this stage accounts for 80 percent of the transforms. To estimate disk usage, consider that each 2kx2k single precision wavefront requires 16 MB space uncompressed, and typically 4 MB compressed. One can also discard the zero padding at this stage and save nearly a factor of 4. To store a seconds worth of simulated data thus requires about 1 GB per emitter. While large, this amount of data is not completely unwieldy considering IDE drives currently sell for a few dollars per GB.

In the second stage, one could choose any telescope aperture and adaptive optics system and propagate the wavefronts generated in the first stage through this system. It should be noted that changing the guide star

brightness is possible after the first stage, though changing the atmospheric model or guide star asterism is not. In this stage each time step depends on the past history, so parallelization cannot be carried out using as many processors. Specifically, it is not particularly effective to carry out fourier transforms on cluster-based supercomputers, as the overhead to transfer data between nodes is comparable to the time required to transform the array. On the other hand, if one wants to find the Strehl ratio over the entire field then at each step of the simulation the wavefront propagation from each beam is independent, and these calculations may be parallelized. Similarly, for MCAO simulations the wavefront propagation from each guide star is independent. Thus, if you do have access to a cluster-based supercomputer with many nodes, you may complete the simulation of many beams simultaneously in the amount of time it would take a single node to simulate a single beam.

6. OBJECT ORIENTED PROGRAMMING

Given the above requirements for the simulation, the resulting software will be fairly complex. First there is the task of representing the different elements of the simulation. These include things like power spectra, wavefront sensors, and emitters. Then there is the problem of specifying the relationships between these elements. Finally, the parallelization layer adds yet more complexity to the software, as these elements must be transferred between processors. In contrast the algorithms that are used in this simulation, such as near and far field wave propagation and matrix operations, are not particularly novel. This indicates that the simulation is organizationally difficult rather than algorithmically challenging. It is the former category for which object oriented languages like C++ are designed.

Object-oriented programming languages encourage design principles that support code reuse. They do so through emphasizing objects and the interactions between these objects rather than the representation of the data. In object-oriented languages, objects are called classes and interactions are typically member functions of these classes. For example, an object-oriented approach to describing a class to represent a wavefront sensor would be to describe its interactions with other classes. Wavefront sensors collect wavefronts and return information about the properties of these wavefronts. These interactions do not depend specifically on the details of the kind of wavefront sensor that is being used. This is an example of the notion of abstraction, in which functionality common to all types of wavefront sensors may be represented through a single interface. Since the specific mechanism through which the wavefronts are processed and analyzed is independent of the interface, these mechanisms may be defined separately.

The C++ programming language supports this separation through the use of inheritance.[11] In this scheme, the interface is specified in an abstract base class - specifically by declaring member functions of this class to be virtual, indicating that the definition of these functions has been deferred. The interface is often referred to as the abstract programming interface (API). This abstract class is then inherited by classes representing concrete realizations of the interface, which must provide definitions for these virtual functions and the data necessary to support this functionality. These are called derived classes. There are several advantages to separating the interface from the implementation through inheritance. First, by emphasizing the interface in this way no assumptions have been made about the data that may reside in the derived class. Second, if these interfaces are used in other parts of the software, then an additional derived classes may be added without the need to change any of this software. Finally, it is typically much easier for another programmer to understand an API than it is to understand the data and operations on this data directly. Because of these reasons, the API plays a much more important role, and a good design is critical for the success of an object-oriented software project.

7. LIBRARY API

Below we present a class hierarchy for a library API that aims to represent the elements of an adaptive optics simulation. Classes and member functions are in boldface type. In this list, inheritance is represented through indentation. Also listed are member functions and virtual member functions that encapsulate the key design philosophy for the library. The notation for these functions are as follows: a member function **f** of a class **A** that takes as an argument an instance **b** of a class **B** and returns an instance of a class **C** is specified as **C A::f(B b)**, or as **C A::f(B & b)** in the case when the argument is passed by reference.

The class planar_surface illustrates an instance of code reuse. This class is inherited by the classes wavefront, plane_optic and detector. Functionality that relates to the specification of a plane surface with respect to a global coordinate system may reside in planar_surface, and this functionality is included in these derived classes through the inheritance relationship.

Within this hierarchy different types of optics inherit the abstract base class optic. The key virtual member function of this class is optic::transform(wavefront), which modifies the wavefront by applying the optical transformation defined in the derived class. While the API contains support only for plane_optics, this abstraction is intended to work for more complex realizations of an optic. The class plane_optic inherits both planar_surface and optic. This is an illustration of multiple inheritance.

The classes wavefront_data and deformable_mirror_commands present a typical use of data hiding through abstraction. There are a number of different forms that wavefront sensor data may take. For example Shack-Hartmann wavefront sensors return the centroids formed by the lenslets in the lenslet array, while curvature sensors return the coefficients in a Zernike expansion of the wavefront phase. However, all wavefront sensor data is used by a reconstructor to make a set of deformable mirror commands. Therefore, all types of wavefront sensor data may be represented using a base class with a virtual member function wavefront_sensor::get_wavefront_data() that returns an instance of a class wavefront_data. This class may then be passed to the function reconstructor::reconstruct(wavefront_data), which returns an instance of the class deformable_mirror_commands. This interaction hides the fact that the actual instance of wavefront_data returned from the wavefront sensor is in fact an instance of one of the derived classes. Similarly, the reconstructor that this wavefront data is passed to is itself actually an instance of one of the classes derived from the reconstructor base class. This example illustrates the strength of the object-oriented design approach.

class pixel_array
A template class to hold a 2d array of objects.
Inherited by:
class pixel_amp_array
This class includes member functions suitable for operations on the real line.
class pixel_phase_array
This class consists of a template instantiation of **pixel_array<float>**
plus member functions suitable for operations on a circle.
key virtual member functions:
void pixel_phase_array::unwrap()
modal_expansion pixel_phase_array::expand()
Note - the class **modal_expansion** is defined below.

class modal_expansion
A base class to represent a modal expansion of a **pixel_phase_array**
key virtual member functions:
pixel_phase_array modal_expansion::phasefront()
Inherited by:
class zernike
class karhunen-loeve

class planar_surface
Represents a planar surface as a point in 3 dimensional space, a vector
normal to the surface, and a rotation angle about the normal vector
Inherited by:
class plane_optic
This class also inherits **optic** - see below.
class wavefront
Class to represent a wavefront as a 2d array of complex field values

key member functions:
void wavefront::geometric_propagator(double distance)
void wavefront::exact_propagator(double distance)
void wavefront::near_field_exact_propagator(double distance)
void wavefront::near_field_fresnel_propagator(double distance)
void wavefront::far_field_fresnel_propagator(double distance)
void wavefront::finite_difference_method_propagator(double distance)

class detector
A base class to represent different types of detectors
key virtual member functions:
void detector::detect(wavefront & wf)
pixel_amp_array detector::readout()
Inherited by:
class infrared_detector
class ccd_detector

class optic
Base class to represent an optic
key virtual member function:
void optic::transform(wavefront & wf)
Inherited by:
class active_optic
class plane_optic
A base class to represent optics that may be approximated as planar surfaces
Inherited by:
class rectangular_lenslet_array
class deformable_mirror
key virtual member function:
void deformable_mirror::update(deformable_mirror_commands)
Note - this member function takes as its argument a class **deformable_mirror_commands** generated by the class **reconstructor**. See below.
class aperture
Inherited by:
class thin_lens
class mirror
class refractive_atmospheric_layer
A class to represent a random phase screen. This class also inherits **active_optic**, and knows how to update itself.

class power_spectrum
A base class to represent different types of power spectra
key virtual member functions:
refractive_atmospheric_layer power_spectrum::get_random_layer()
structure_function power_spectrum::get_structure_function()
See below for a description of the class **structure_function**.
Inherited by:
class power_law_spectrum
A class to represent different types of analytically specified power spectra with power law dependencies and optional inner and outer scales.
class generic_power_spectrum
A class to represent a numerically specified power spectrum

class structure_function
A class to represent a structure function.
key virtual member functions:
void structure_function::add_staticstics(refractive_atmospheric_layer & ref_atm_layer)

class refractive_atmospheric_model
A class to represent a model of the refractive atmosphere
as an array of power spectra at different heights.

class optical_system
A class to hold an ordered array of optics, and positional information on this system.
key member functions:
void optical_system::insert(optic & op)
void optical_system::insert(optical_system & op_sys)
void optical_system::set_geometric_propagation()
void optical_system::set_diffractive_propagation()
void optical_system::propagate_forward(wavefront & wf)
void optical_system::propagate_backward(wavefront & wf)
void optical_system::step_forward(wavefront & wf)
void optical_system::step_backward(wavefront & wf)
Inherited by:
class refractive_atmosphere
A class to represent the refractive atmosphere, consisting of an array of
refractive_atmospheric_layer objects and their heights above the ground
key virtual member functions:
refractive_atmosphere::refractive_atmosphere(refractive_atmospheric_model)

class emitter
A class to represent a source of wavefronts.
key virtual member function:
wavefront emitter::emit()
Inherited by:
class plane_wave_emitter
class spherical_wave_emitter
class extended_emitter

class wavefront_sensor
A class to represent wavefront sensors
key virtual member functions:
void wavefront_sensor::detect(wavefront & wf)
void wavefront_sensor::convolve(extended_emitter & ext_emtr)
wavefront_data wavefront_sensor::readout()
Note: this member function returns an instance of a class **wavefront_data**,
which is used by the class **reconstructor**. See below.
Inherited by:
class shack_hartmann_wavefront_sensor
class shearing_interferometer_wavefront_sensor
class curvature_wavefront_sensor
class pyramid_wavefront_sensor
class layer_oriented_wavefront_sensor

class wavefront_data
A base class to hold different types of wavefront data.
Inherited by:
class shack_hartmann_data
class shearing_interferometer_data
class curvature_data
class pyramid_data
class layer_oriented_wavefront_data

class deformable_mirror_commands
A base class to hold different types of deformable mirror commands.

class reconstructor
A base class to represent the reconstructor.
key virtual member functions:
deformable_mirror_commands reconstructor::reconstruct(wavefront_data)
Inherited by:
class Iterative reconstructor
Inherited by:
class Jacobi reconstructor
class Gauss_Seidel_reconstructor
class multigrid_reconstructor
class successive_overrelaxation_reconstructor
class least_squares_reconstructor
class maximum_likelihood_reconstructor
class maximum_a_priori_reconstructor

REFERENCES

1. V. I. Tatarskii, *The effects of the turbulent atmosphere on wave propagation*, Jerusalem: Israel Program for Scientific Translations, 1971.
2. R. Avila, J. Vernin, and S. Cuevas, "Turbulence Profiles with Generalized SCIDAR at San Pedro Mártir Observatory and Isoplanatism Studies," *Publications of the Astronomical Society of the Pacific* **110**, pp. 1106–1116, Sept. 1998.
3. R. G. Lane, A. Glindemann, and J. C. Dainty, "Simulation of a komolgorov phase screen," *Waves in Random Media* **2**, pp. 209–224, 1992.
4. J. M. Martin and S. M. Flatte, "Simulation of point-source scintillation through three-dimensional random media," *J. Opt. Soc. Am. A* **7**, pp. 838–847, May 1990.
5. J. W. Goodman, *Introduction to Fourier optics*, Introduction to Fourier optics, 2nd ed., Publisher: New York, NY: McGraw-Hill, 1996.
6. M. Born and E. Wolf, *Principles of optics. Electromagnetic theory of propagation, interference and diffraction of light*, Oxford: Pergamon Press, 1980, 6th corrected ed.
7. R. J. Sasiela, *Electromagnetic wave propagation in turbulence. Evaluation and application of Mellin transforms*, Springer Series on Wave Phenomena, Berlin: Springer, —c1994.
8. E. A. Sziklas and A. E. Siegman, "Mode calculations in unstable resonators with flowing saturable gain. 2: Fast Fourier transform method," *Applied Optics* **14**, pp. 1874–1889, Aug. 1975.
9. M. V. Papalexandris and D. C. Redding, "Calculation of diffraction effects on the average phase of an optical field," *J. Opt. Soc. Am. A* **17**, pp. 1763–1772, 2000.
10. R. N. Bracewell, *The Fourier Transform and its applications*, McGraw-Hill Series in Electrical Engineering, Networks and Systems, New York: McGraw-Hill, —c1986, 2nd rev.ed.
11. B. Stroustrup, *The C++ Programming Language*, Addison Wesley. Reading Mass. USA., 2000.