Arroyo

Matthew C. Britton

Matthew C. Britton, "Arroyo," Proc. SPIE 5497, Modeling and Systems Engineering for Astronomy, (16 September 2004); doi: 10.1117/12.552316

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2004, Glasgow, United Kingdom

Arroyo

Matthew C. Britton

California Institute of Technology, Pasadena, CA 91125 USA

ABSTRACT

Arroyo is an open source, cross-platform C++ class library project designed for modeling of electromagnetic wave propagation through atmospheric turbulence and adaptive optics systems. This paper describes the functionality available in the library and discusses future plans for this project.

Keywords: Adaptive optics, telescopes, diffraction, simulation, turbulence

1. INTRODUCTION

Astronomical adaptive optics is a dynamic field of research, with new architectures, algorithms, and applications being proposed at a remarkable pace. Sophisticated techniques have been suggested for sensing atmospheric turbulence, such as the use of multiple guide stars for sensing the three dimensional distribution of turbulence[1] and focal plane wavefront sensing for performing high precision phase compensation.[2] Novel instrument designs have been proposed for applications ranging from wide field diffraction-limited spectroscopy to very high contrast imaging. There are also numerous proposals for performing wavefront reconstruction and control, and for postprocessing techniques such as field dependent point spread function (PSF) reconstruction from adaptive optics telemetry data.[3] There are significant challenges in analyzing these adaptive optics architectures and their accompanying instrumentation. Unlike the current generation of general purpose single conjugate adaptive optics systems, these proposals illustrate a trend towards optimizing the adaptive optics system architecture and algorithms for particular types of science. From this perspective, the traditional adaptive optics metric of residual wavefront error is often no longer appropriate. One would instead like to evaluate performance in terms of more relevant scientific metrics such as astrometric precision, photometric precison, or the contrast of the science image.

Given the growing complexity of these system architectures and and the expense of their implementation, time domain numerical simulations[4,5] can play a very useful role in providing a quantitative evaluation of their capabilities. By generating random realizations of atmospheric turbulence and performing wave propagation through this turbulence, a simulation can model the effects of turbulence degradation, and can preserve correlations between beams that arise from tilt and focal anisoplanatism. A simulation can accurately represent the effects of scintillation, which are important in high precision applications such as direct planetary imaging and in atmospheric monitoring using scidar. With models of the adaptive optics system components and by replicating the reconstructor and control law, a simulation can accurately represent the process of wavefront sensing and correction. Such functionality may be used to generate performance predictions for an adaptive optics system, and in tolerancing the system design. The highly controlled environment provided by simulation affords the opportunity of performing quantitative comparisons between different reconstruction and control algorithms. Finally, a simulation can generate field dependent PSFs in the science focal plane, which may be used to evaluate scientific performance metrics other than residual wavefront error. And together with a simulated set of telemetry data, these PSF's may be used to test PSF reconstruction algorithms.

The simulation functionality described above is applicable to a broad range of problems both within and outside astronomical adaptive optics. This suggests an approach in which the simulation problem is factored, so that common functionality is implemented once and then reused in the development of many different applications. Such an approach has the obvious advantage of avoiding duplication of effort. It also has the advantage that many applications rely on the same code, so that validation is more thorough and errors are more quickly

Further author information: Send correspondence to Matthew Britton: E-mail: mbritton@astro.caltech.edu

Modeling and Systems Engineering for Astronomy, edited by Simon C. Craig, Martin J. Cullum, Proc. of SPIE Vol. 5497 (SPIE, Bellingham, WA, 2004)
0277-786X/04/$15 doi: 10.1117/12.552316

discovered. This paradigm is one that is often used in software development. Reusable functionality is incorporated into a library, which is then distributed to users who can employ this functionality in writing programs specific to their application.

Arroyo is a C++ software library that aims to provide functionality for the simulation of electromagnetic wave propagation through atmospheric turbulence and adaptive optics systems. Its library nature has the dual advantage of hiding the complexity of this functionality behind the library interface and permitting reuse of this functionality in many different applications. Arroyo's C++ implementation allows users to extend the class hierarchies within the library through inheritance and templatization. The library is designed to be portable across platforms. It is also possible to address computationally challenging simulation problems by using the library in multithreaded and distributed computing applications. To facilitate the widespread use of this functionality, the source code has been released under the Gnu Public License. This paper describes the status and future plans for the Arroyo project.

2. LIBRARY FUNCTIONALITY

Arroyo's library functionality is implemented using object oriented programming paradigms.[6] The library design is based on a set of class hierarchies, each of which represents a category of objects. Optics are an example of such a category. These class hierarchies are extended through derivation to form concrete realizations. Interactions between classes are implemented using virtual functions, which are overloaded in the derived classes. This section discusses Arroyo's major class hierarchies and their interrelationships.

2.1. Geometry

Many of the applications that Arroyo aims to address require the representation of geometric relationships among many different objects. One example is in wave propagation through the turbulent atmosphere from sources at different locations in the sky. The correct locations and orientations of wavefronts from each of these sources must be represented to accurately model the effects of anisoplanatism. Arroyo employs a system loosely based on techniques of coordinate free geometry.[7] The geometric concepts of points, vectors, and reference frames are each encoded in a class. Geometric relationships between instances of these classes are available through the library interface. Examples include computing the distance between two points or finding the coordinate of a point in a particular reference frame. These classes are then inherited by classes whose realization requires a geometric definition. For example, a wavefront requires a reference frame to define its location and orientation in three dimensional space. Arroyo also contains a class hierarchy that permits transformations of geometric objects, with derived classes for translations, rotations, reflections and scalings. By design, Arroyo's geometric functionality does not require the selection of any preferred frame in the simulation, as the library interface only permits relative calculations between geometric objects.

2.2. Emitters and Wavefronts

Arroyo contains a class hierarchy that is used to represent emitters, which may be regarded as sources of electromagnetic wavefronts. Concrete examples include emitters of spherical and plane waves. The geometric properties of these classes are defined by inheriting a point and a vector, respectively. In the future, Arroyo will support emitters with finite spatial extent, which may be two or three dimensional. As examples, the former may be used to represent extended science objects and the latter can represent extended backscatter from laser beacons. As a future functional extension, these classes will also contain information on the electromagnetic spectra of the sources they represent.

Arroyo contains a class to represent wavefronts. Internally, instances of this class consist of a two dimensional array of complex numbers that represent the amplitude and phase of the wavefront, along with some additional header information. This header information records the pixel dimensions of the complex array, the pixel scale, the geometric location and orientation of the wavefront in three dimensional space, a timestamp, and the wavelength of the electromagnetic radiation. The overall spherical curvature of the wavefront is also stored separately. For wavefronts with even moderate curvature, this technique permits the removal of many phase wraps from the wavefront phase, improving the fidelity of the representation. Collectively, all the information except the two dimensional array of complex numbers forms a wavefront header, which itself is a class. This class

permits operations that involve only the header information to be performed efficiently. An example is discussed below.

There are several different ways to instantiate a wavefront, but one of these is of particular utility in many applications. In this approach, one requests a wavefront from an emitter by supplying the wavefront header information to a member function of the emitter class instance. This member function uses the electromagnetic wavelength, geometric location, pixel dimensionality, and pixel scale to determine how the complex array should be initialized.

Given an instance of a wavefront, one can propagate this wavefront through free space using a number of free space propagators available as member functions of the class. Specifically, Arroyo provides implementations of the exact propagator and a geometric propagator suitable for near field propagation. Arroyo also provides four paraxial propagators: near field angular and Fresnel propagators, and far field Fresnel and Fraunhoffer propagators.[8] These propagators are implemented using fast Fourier transforms. The near field paraxial propagators preserve the wavefront pixel scale, while for the far field paraxial propagators the final pixel scale is inversely proportional to the initial one. An alternative discrete Fourier transform implementation of the far field paraxial propagators is also included in the library. These implementations are based on the Goertzel-Reinsch recursion relation,[9,10] and permit selection of arbitrary sampling and array size in the final wavefront. It should be noted that the wavefront instance does not contain enough information to choose whether to employ a near or far field propagator to perform the propagation. Part of the design criteria for this class is that it does not need to record its past history. Therefore only the calling routine is in a position to know information such as the Fresnel number. All of these propagators increment the timestamp in the wavefront by the propagation distance divided by the speed of light, and update the geometric location of the wavefront.

2.3. Optics

In addition to propagating through free space, one would like to be able to transform a wavefront using various optics. Optics form one of the largest class hierarchies in Arroyo. Currently all optics in Arroyo are represented by two additional base classes representing plane and one to one optics. The former are optics whose geometric properties may be represented by a three dimensional reference frame. Examples include an aperture, a thin lens, and a deformable mirror. One to one optics have the property that they accept one wavefront as input and return a single wavefront as output. All the examples of plane optics listed above are also one to one optics. An example of a one to many optic would be a pyramid sensor, which splits one wavefront into four separate wavefronts, each propagating in a different direction.

One to one optics have a virtual member function called transform, which takes a wavefront as an argument. This member function modifies the wavefront in a way that represents the effect of the optic. For apertures, this effect would be to zero elements of the underlying two dimensional complex array that are masked by the aperture. For a deformable mirror, the effect would be to modify the wavefront phase to account for the relative pathlengths introduced by the actuators, and to reorient the propagation direction of the wavefront according to the law of reflection. In this case, the transformation will depend on the wavelength of the radiation. Since wavefronts and plane optics understand their location and orientation in three dimensional space, one may effect transformations for wavefronts that are decentered or incident at an angle to the optic. The details of these transformations are defined in the the transform member function overloaded by the derived class. Specific classes that reside in the optic class hierarchy are discussed in more detail below.

2.4. Atmospheric Turbulence

Random realizations of atmospheric turbulence play a central role in simulations of adaptive optics systems. In Arroyo the functionality to generate these random realizations is factored into a number of different classes. The library contains a class for representing power spectral statistics of atmospheric turbulence. This class can represent an isotropic power law with arbitrary exponent and coefficient, and optionally an exponential or Frehlich inner scale and a von Karman or Greenwood outer scale.[11] Arroyo contains a class to represent two dimensional random atmospheric phase screens. Instances of this class may be constructed using a particular power spectrum and subharmonic method,[12,13] and both the pixel scale and dimensions of the screen are free

parameters. This class is a type of one to one plane optic, and can transform an incident wavefront by adding its aberrations to the wavefront phase. An example of one such phase screen is shown in Figure 1.

Arroyo contains a class hierarchy to represent multilayer atmospheric models. From a practical standpoint one can consider this class to contain information on the heights, weights, and power spectral statistics of each of the layers in the model. One can construct such a model by specifying this information, or one may use one of the specific atmospheric models available in the library. Examples include nighttime models of mean turbulence profiles at Cerro Pachon[14] and Mauna Kea,[14] the Hufnagel Valley model,[11] and the SLCSAT day and night models.[11] Each of these models has a different parameterization, but one is generally specifying the overall Fried parameter and relative weights of the layers either implicitly or explicitly. Arroyo also contains a class hierarchy for representing models of the vertical wind profile. Currently Arroyo contains only one such model,[15] which contains a tropospheric and ground layer wind component. This wind model is used in selecting random wind velocities for each of the layers in the atmospheric model in a way that enforces vertical correlations in these velocities. The atmosphere and wind models are used together to generate the random turbulence screens that are employed in the simulation. These screens are blown past the telescope aperture in order to model the dynamical evolution of turbulence. Currently the screens evolve under the assumption of frozen flow, and no internal dynamical evolution is modeled.

For many applications the atmospheric phase screens dominate the memory reqirements for the simulation, and for large problems can often exceed the two Gigabyte memory limitation on 32 bit processors. It is thus important to make these screens as small as possible. Clearly the size of the screens that one must construct depends on the geometry of the problem one wishes to simulate, as these screens must be large enough so that wavefronts from all the emitters spread over the field of view can propagate down through these screens to the telescope aperture. The minimum screen size depends on the particular field of view, the telescope aperture diameter, the layer heights in the atmospheric model, and the duration of the simulation. Less obviously the minimum size depends on the electromagnetic wavelength of the radiation one wants to propagate if one wishes to use a near field propagator. This dependancy arises because the wavefront is aperiodic after transformation by the atmospheric phase screen, and the Fourier transformations used to effect near field propagation corrupt the edges of the wavefront. The region of corruption increases with propagation distance and decreases with the wavelength of the radiation. To obtain uncorrupted data at the aperture, one must propagate a wavefront large enough so that this corruption does not leak into the region of the wavefront accepted by the aperture.

In Arroyo, constructing layers of minimal size is a two step process. First, one would like to establish the minimum dimensionality of a wavefront that will completely cover the aperture that one wishes to use in the simulation. This must take into account the near field propagation technique. To compute the minimum size of this wavefront, one calls a member function of the atmospheric model class that takes as arguments the emitter that will be used as a source of the wavefronts, the wavelength and pixel scale of the wavefront, the aperture to which the wavefront will be propagated, and the propagation technique. This function returns a wavefront header that is located at the highest atmospheric layer in the refractive atmospheric model and is oriented along the line joining the emitter to the center of the aperture. The wavefront header also has been initialized with the minimal dimensionality that will suffice to deliver valid wavefront data at the aperture given the requested propagation technique.

Having accumulated wavefront headers for each of the guide stars and science targets at all wavelengths that one wishes to simulate, one would like to generate atmospheric phase screens of minimal size. To do so, one calls another member function of the atmospheric model class, which takes as arguments wavefront headers from each of the guide stars and science targets and the duration of the simulation. Within this member function, random numbers are drawn to determine the wind velocities of the layers. This function computes the minimum dimensions of the screens required to perform the simulation. Then many more random numbers are drawn to generate these screens. The function returns an array of phase screens, each minimally sized according to its height, velocity, and pixel scale. An important feature of Arroyo is its ability to reproduce the same phase screens by initializing the random number generator with the same seed, assuming all other parameters that control the size of these screens are unchanged. This permits controlled comparisons between different adaptive optics systems.

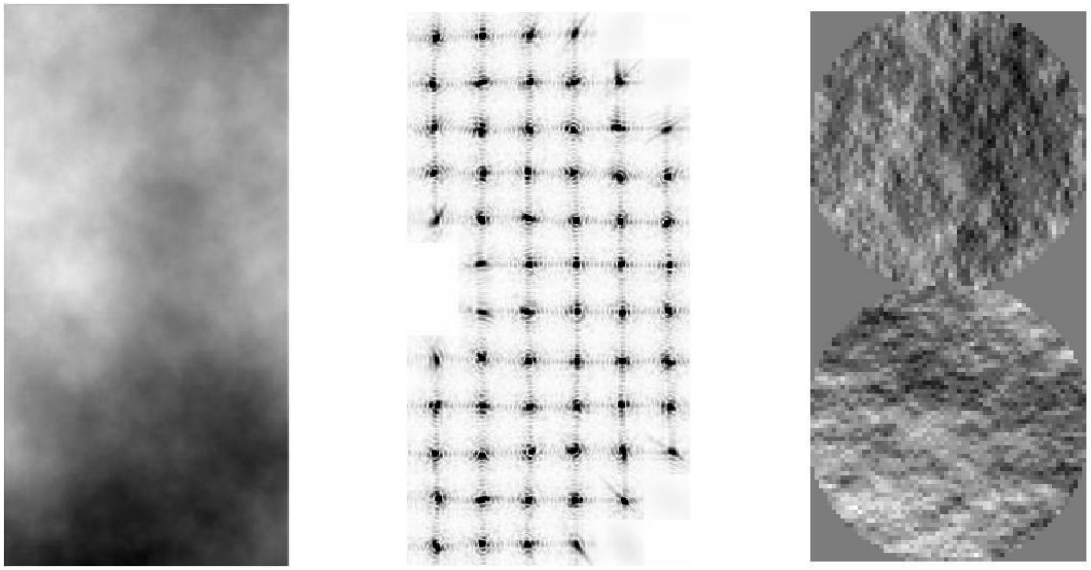

Figure 1. Functionality available in Arroyo. An example of a random phase screen is shown on the left. The center image displays one half of an image formed by a 12x12 lenslet array that images turbulence corrupted wavefronts through an annular aperture. The PSF formed by each lenslet displays the diffraction pattern arising from propagation through a square aperture, while partially illuminated subapertures at the edge of the telescope aperture generate elongated spots. Distortions arising from atmospheric turbulence are also visible. The image on the right shows centroids calculated from a 64x64 lenslet array imaging turbulence corrupted wavefronts through a circular aperture. The x and y centroids are shown in the lower and upper halves of this figure, respectively.

2.5. Apertures

Apertures are another example of a plane, one to one optic. Currently Arroyo supports rectangular, circular, annular, spidered annular, hexagonal, and segmented hexagonal apertures. These apertures do not store pixellated arrays of ones and zeros. Instead, they store minimal information about the aperture dimensions. For example, a spidered annular aperture stores the inner and outer diameters of the annulus, the number of spiders, and the width of the spiders. The transform member function is overridden by these classes to mask the wavefront. For each pixel in the two dimensional wavefront, this function computes whether the pixel lies within or outside the aperture. For wavefront pixels that straddle the aperture boundary, the member function weights their amplitude by a factor equal to the fraction of the pixel lying within the aperture. This is a particularly important feature for modeling segmented hexagonal apertures, where the gap size between hexagonal segments is typically much smaller than the pixel scale of the wavefront. In this way, Arroyo can generate PSF's for segmented hexagonal apertures using a diffractive far field propagator. This is an alternative to the gray pixel approximation.[16]

Apertures may be used to define the reflective surface of the telescope primary, but are also used to define the reflective surface boundaries of other optics as well. For example, one can specify the clear surface of a tip tilt mirror using an aperture.

2.6. Lenslet Arrays and Centroids

Arroyo contains a class for representing lenslet arrays, which form another example of plane, one to one optics. Currently only square subapertures are supported. This class is instantiated by providing a lenslet pitch and a focal length, along with the number of lenslets across the array. Transformation of the wavefront is performed individually for each lenslet. This approach permits one to downweight wavefront pixels that straddle lenslet boundaries by the fractional areal overlap. For each lenslet, the local region of the wavefront is extracted and the lenslet curvature is added to the wavefront phase. A far field Goertzel-Reinsch propagator is used to compute the contribution of each lenslet to the final wavefront in the focal plane. This propagator allows specification of

arbitrary pixel scale and array dimensions in the focal plane, and this freedom is employed to allow the user to choose the number of pixels per lenslet and the region of support in the focal plane. By choosing the region of support to be larger than the lenslet pitch, cross-coupling between lenslets may be modeled. This procedure is repeated for each lenslet separately, and the resulting wavefronts are recombined coherently in the focal plane. An example of a wavefront in the focal plane of a lenslet array is shown in Figure 1.

Arroyo contains a class for representing centroids. Since detectors are not yet supported by Arroyo, there is a simplified procedure for measuring centroids directly from wavefronts formed by lenslet arrays. This constructor uses a classical centroiding algorithm to compute the centroids from a wavefront in the lenslet array focal plane. An example of these centroids appears in Figure 1.

2.7. Deformable and Tip Tilt Mirrors

Arroyo contains a class hierarchy for representing tip tilt mirrors. Currently there is only one derived class in this hierarchy. This class implements a model of a tip tilt mirror as a plane, one to one optic whose clear surface is defined by one of Arroyo's aperture classes. The model also includes a parameter that is used to specify the angular velocity at which the mirror moves. This class contains a member function that updates the orientation of the mirror, which takes as an argument the orientation to which to drive the mirror and the time at which this command was issued. Arroyo currently employs a simple model for the mirror response. When the update member function is called, the mirror begins to move instantaneously at the specified angular velocity. If it reaches its commanded position before the next update command is issued, it stops instantaneously. If not, it changes direction instantaneously and starts moving at the same rate towards the new commanded position. Higher fidelity dynamical models may be added to Arroyo in the future. The transform member function is implemented for this class using the law of reflection, and masks the wavefront according to the aperture used to define the clear surface of the mirror.

Arroyo contains a class hierarchy for representing deformable mirrors. This hierarchy currently contains a single derived class, whose implementation is similar to that of the tip tilt mirror class described above. The model treats the deformable mirror as a plane, one to one optic whose clear surface is defined by one of Arroyo's aperture classes. The model supports rectilinear arrays of actuators, and uses a pyramidal influence function for the actuators. This class does not model actuator hysteresis or gain variations. The model also contains a parameter that is used to specify the linear velocity at which each actuator moves. This class contains a member function to update the actuator locations, which takes as an argument the locations to which to drive the actuators and the time at which this command was issued. When the update member function is called, each actuator begins to move instantaneously at the linear velocity specified in the constructor. If an actuator reaches its commanded position before the next update command is issued, it stops instantaneously. If not, it instantaneously starts moving at the same rate towards the new commanded position. There is some evidence to support this model for the case of PMN actuators.[17]

The implementation of the transform member function for the deformable mirror is somewhat more complex than that for the tip tilt mirror. The deformable knows its own geometric location and orientation as well as those of the wavefront. To effect the transformation, the location of each wavefront pixel is computed relative to the surface of the deformable mirror. Based on the timestamp in the wavefront header, the DM calculates the locations of all the actuators and computes the correction to be added to the wavefront phase. Finally, the transform member function applies the law of reflection and masks the wavefront according to the aperture used to define the clear surface of the mirror.

2.8. Reconstruction and Control

Arroyo contains a class hierarchy for representing single conjugate reconstructors. These reconstructors are used to reconstruct deformable mirror and tip tilt residuals from sets of centroids. Currently three different derived classes reside in this reconstructor hierarchy. The first supports generation of least squares reconstructors. Such reconstructors depend on the relative positions and orientations of the aperture, lenslet array, and deformable mirror, and reconstructors may be generated for arbitrary geometries. The implementation assumes a pyramidal influence function to compute the geometry matrix, which relates actuator displacements to centroid measurements. The calculation is effected by computing the intersection of each facet of the pyramidal influence function

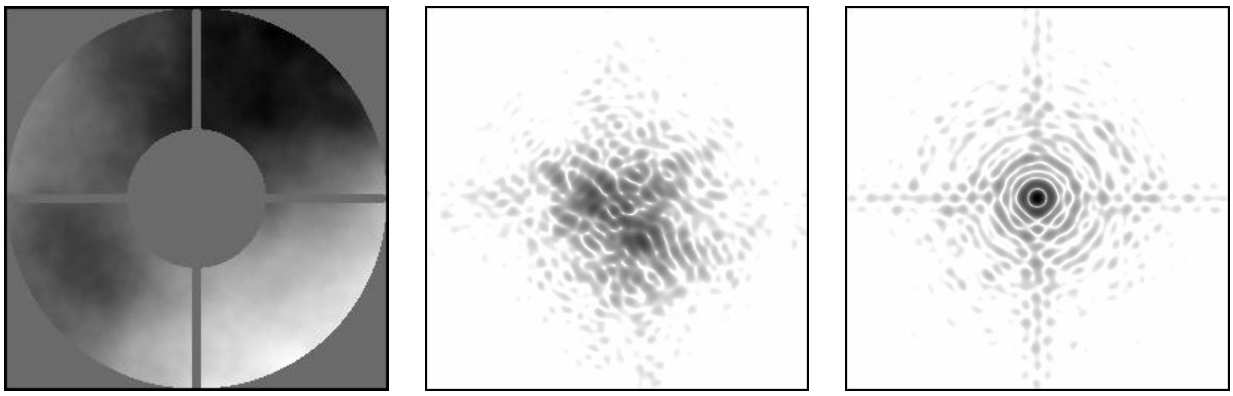

Figure 2. Example of a closed loop adaptive optics simulation for a 16x16 actuator Shack Hartmann adaptive optics system on a 5 meter telescope. The simulation was performed using a six layer Cerro Pachon atmospheric turbulence model,[14] with an r_0 of 15 cm. The left panel shows the uncompensated wavefront phase in the pupil plane. An annular aperture with four spiders was assumed. The center panel shows the uncompensated PSF. The right panel shows the PSF after compensation by the adaptive optics system. The images were computed over a four arcsecond field oversampled by a factor of 8 relative to Nyquist, and are displayed on a log stretch.

for each actuator in the deformable mirror with each subaperture of the lenslet array, and relies on an algorithm from computational geometry for finding the intersection of two convex polygons.[18] Once this region has been defined, it is straightforward to compute the integral of the partial derivatives of the pyramidal influence function over this region. Once the geometry matrix has been computed, the reconstructors are formed through least squares inversion of the this matrix using singular value decompositon (SVD). In addition to this class, Arroyo also contains two reconstructor classes that can instantiate themselves by reading from file reconstructors generated outside the library. The first reads reconstructors generated by the program A++, while the second reads reconstructors used by the Palomar adaptive optics system PALAO.

Arroyo contains a class that encapsulates the functionality of proportional integral control. This class takes the deformable and tip tilt mirror residuals and converts them to command vectors. The user can select the proportional and integral gains when instantiating this class.

2.9. Performing Closed Loop Adaptive Optics

The section concludes with an explanation of how the classes described above may be used to perform a closed loop single conjugate adaptive optics simulation. Variations on the procedure below are relatively straightforward, as calls to the library may be rearranged within the application to modify the simulation. Generation of the atmospheric phase screens yields wavefront headers for each of the guide stars and science targets. For each iteration of the simulation, each wavefront header is supplied to its respective emitter to generate a wavefront, which is located at the highest atmospheric phase screen and is directed towards the center of the telescope aperture. The screen's transform member function is called to transform the wavefront. This member function uses the timestamp in the wavefront header and the wind velocity of the screen to determine where the center of the wavefront hits the screen. Using the geometric information contained in the wavefront and knowing its own geometric location, the screen computes the location of each wavefront pixel relative to those of the layer. For each wavefront pixel, the member function computes the areal overlap with each layer pixel and adds in the corresponding optical path difference divided by the wavelength of the wavefront, weighted by the areal overlap. Once this transformation is complete, one calculates the distance to the next layer along the direction of propagation and performs a free space wave propagation to reach the next layer. This propagation may be geometric or diffractive depending on the technique the user wishes to employ. This procedure process is repeated until the wavefront reaches the aperture of the telescope. These steps can be performed in reverse to model laser guide star uplink propagation.

At the ground, the wavefronts are transformed by the aperture. Both sensing and science wavefronts are then transformed by the deformable and tip tilt mirrors. The science wavefronts are propagated to the far field where they are accumulated to form the science PSF's over the field of view spanned by the science emitters. The sensing wavefront is passed through the lenslet array, where it is used to construct the Shack Hartmann centroids. These centroids are then passed to the reconstructor, which generates tip tilt and deformable mirror residuals. The residuals are in turn passed to the proportional integral controller, which outputs commands used to update the mirrors. The entire procedure is repeated in a loop over timesteps. An illustration of such a closed loop simulation is shown in Figure 2. The compute time required to perform closed loop adaptive optics simulations on apertures ranging from 3.5 to 30 meters in diameter is shown in Figure 3.

Arroyo contains the important feature that none of the classes require updates to occur on regular time intervals. Wavefronts can be timestamped using a continuous timeline, and the atmospheric phase screens and the deformable and tip tilt mirrors can transform wavefronts with arbitrary timestamps. The mirrors can also be updated at any time. This permits the representation of arbitrary event timelines. This feature represents an advantage for certain applications, such as simulations of adaptive optics systems containing multiple control loops with independent update rates, and of laser guide star systems that aim to employ range gating schemes to suppress Rayleigh backscatter from multiple beacons.

3. PROJECT STRUCTURE

Arroyo aims to employ modern software development techniques, and contains a number of important features not directly related to the simulation functionality. The source code for Arroyo has been publically released under the Gnu Public Licence. The software is distributed with a substantial amount of documentation. Cross-linked HTML pages are automatically generated during compilation, which include inheritance graphs for the class hierarchies and member function documentation for each class. A tutorial is also distributed with Arroyo. The distribution contains about twenty example programs. Some of these programs contain a small amount of code that aims to explore a particular element of the library functionality. Other programs represent highly parameterized simulations of single conjugate adaptive optics systems.

The Arroyo distribution comes with a set of regression testing routines that can be used to validate the library functionality. Arroyo has been successfully compiled and tested on a variety of platforms, including Solaris, Cygwin and Linux. Arroyo has also been installed on 64 bit AMD processors running Suse and Mandrake distributions of the 64 bit Linux operating system. Arroyo's library functionality has been used in multithreaded and distributed computing applications. An example of the former is included in the distribution.

Arroyo employs the Fits file format for reading and writing to disk, and almost all classes in the library have a file format. Writing a class instance is as simple as calling a member function called write, which takes as a single argument the filename. This member function is reimplemented in every class, so that a specific file format is written for each type of object. In this way the user has complete control over what data is written out during a simulation. For instance, in the example program that simulates a closed loop single conjugate adaptive optics system, the user may output wavefronts in the pupil plane at both the wavefront sensing and science wavelengths and science PSF's along with full adaptive optics telemetry data at the control loop rate. This telemetry data includes deformable mirror and tip tilt mirror commands and residuals, wavefronts in the lenslet focal plane, and the computed centroids. Each class that has a file format also contains a member function called read, which can load a Fits file containing an instance of the class. Polymorphic instantiation of classes from file is supported through the Object Factory paradigm.[19] In this paradigm, an instance of the derived class that is stored within the file is constructed dynamically and a base class pointer is returned to the calling routine. This paradigm permits Arroyo users to write library extensions through inheritance and have applications recognize the file formats for these derived classes.

Compilation of Arroyo requires three third party libraries. CFITSIO[20] is used to read and write fits files. The Fastest Fourier Transform in the West[21] (FFTW) is used to perform the fast Fourier transforms required in several parts of the library. Finally, LAPACK[22] is used to perform the SVD employed in generating Arroyo's single conjugate reconstructors. The source code for CFITSIO and LAPACK is publically distributed without restriction, while FFTW is released under the Gnu Public License.

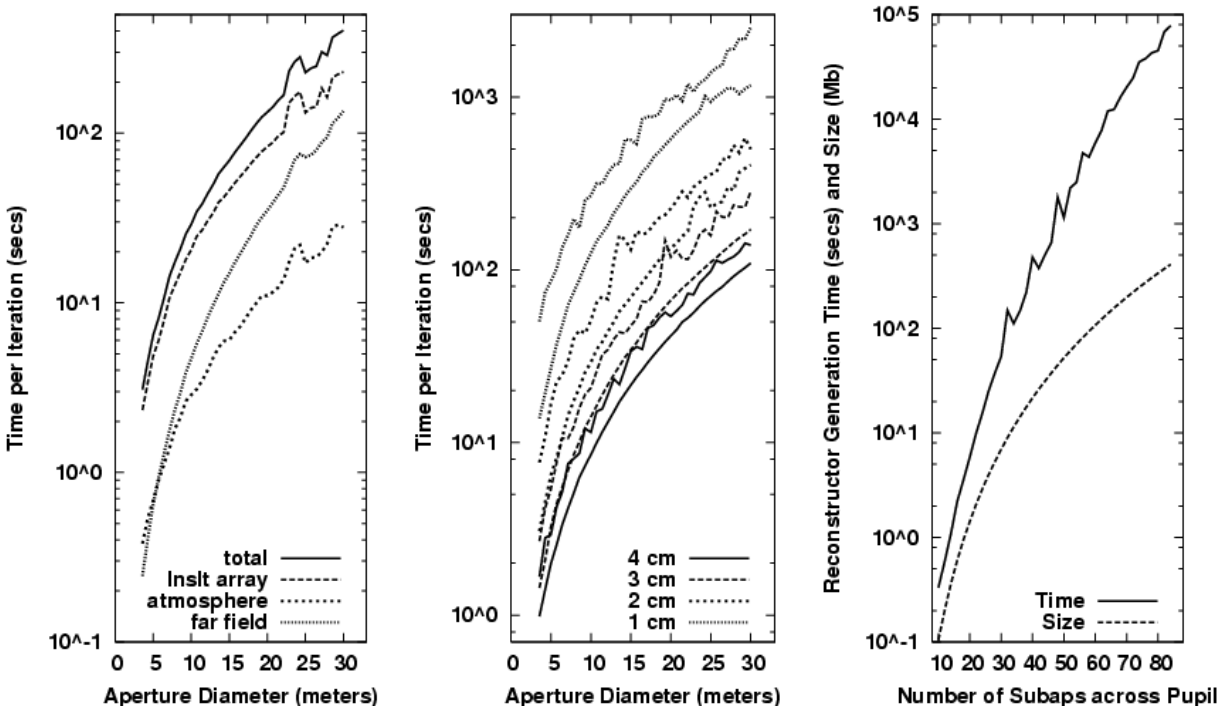

Figure 3. Compute time for single conjugate adaptive optics simulations on a 2.4 GHz Pentium 4. These simulations were performed using a six layer Cerro Pachon atmospheric turbulence model.[14] A fixed subaperture size of 35.7 cm was used, and the number of subapertures across the pupil ranged 10 to 84 for aperture diameters from 3.6 to 30 meters. Monochromatic radiation at .5 μm was used for wavefront sensing, and at 2.2 μm to simulate the science image. The science image was calculated over a 4 arcsecond field of view and was oversampled by a factor of 8 relative to Nyquist. The left panel shows the total time required per iteration of the control loop. In these simulations, wavefronts and atmospheric screens were sampled at a pixel scale of 2 cm and were propagated through the atmosphere using a geometric propagator. Also shown are the largest three computational components: propagation through the lenslet array, propagation through the atmosphere, and propagation to the far field. With the field size in arcseconds and the sampling defined relative to Nyquist, the last component is increasing with aperture diameter as the diffraction limit decreases. The central panel shows the total simulation time per iteration of the control loop for pixel scales between 1 and 4 cm. Two curves are plotted for each pixel scale, corresponding to geometric and diffractive propagation through the atmosphere. The right panel shows the time required to compute the least squares reconstructor and the space required for its storage.

4. FUTURE FUNCTIONALITY

This section describes some of the functional extensions planned for Arroyo. As yet Arroyo does not contain a detector class. One would want an instance of such a class to sit in the focal plane of a lenslet array or a science camera to detect the wavefront. Broadband observations could be synthesized by detecting multiple wavefronts with a range of wavelengths. The detector would understand the spectral evolution of its quantum efficiency, and could appropriately weight the wavefronts according to its responsivity. One could then read out the detector to obtain an image. The detector would understand how to add its read noise, dark current and photon noise into this image. Due to the lack of a detector class, Arroyo does not support simulations that accurately represent absolute photometry. Arroyo also lacks support for sodium layer models and extended emitters. This functionality is required to accurately model sodium laser guide star spot elongation.

Arroyo does not contain support for single conjugate reconstructor algorithms other than the least squares reconstructor described above. In the future, optimal, predictive and fast reconstructor techniques may be added to the library. This functionality would permit quantitative comparisons of these reconstructor algorithms against least squares techniques. Arroyo does not yet include support for tomographic reconstruction algorithms used in multiconjugate adaptive optics systems.

There are a number of types of optics that could be added to the library to support modeling of diffractive propagation through optical systems. One example would be a conic mirror class, which would be able to transform wavefronts by raytracing past the surface of the conic. Aggregate classes formed from multiple conic mirrors could be used to represent segmented mirror optics. Other possible extensions include dichroics and pyramid sensors. Functionality like this would permit accurate modeling of diffractive wave propagation through optical systems. This type of modeling may be useful for calibration algorithms such as phase diversity, and applications like direct imaging of planetary companions that require extremely high precision adaptive optics systems.

5. CONCLUSIONS

Arroyo emphasizes the use of modern software development techniques, employs an object oriented design, and follows an open source model. The design of Arroyo as a library permits the reuse of library functionality in a diverse range of applications. Implementation of this library in C++ affords the opportunity of extending the functionality to develop new models for system components and to implement new algorithms. These features makes Arroyo an attractive simulation tool, and the project has a substantial and growing user base. Applications have included investigations on the effects of partially compensated turbulence on coronography and on segmented mirror telescopes, and in the simulation of satellite to ground communication links. This user base plays an important role in maintaining bug-fee code, porting the library to new platforms, and in directing the development of future library functionality. While it is too early to judge the long term sustainability of the project, there are encouraging signs that the library will appeal to a relatively broad audience, both within and outside of astronomy.

ACKNOWLEDGMENTS

The author acknowledges the contributions of the many colleagues who have offered advice and support in the development of Arroyo. Richard Dekany, Mitchell Troy, Don Gavel and Jose Milovich deserve particular mention. The author thanks Alan Morrissett for contributing to the assembly of this project into a formal release.

The Thirty Meter Telescope (TMT) Project is a partnership of the Association of Universities for Research in Astronomy (AURA), the Association of Canadian Universities for Research in Astronomy (ACURA), the California Institute of Technology and the University of California. The partners gratefully acknowledge the support of the Gordon and Betty Moore Foundation, the US National Science Foundation, the National Research Council of Canada, the Natural Sciences and Engineering Research Council of Canada, and the Gemini Partnership.

REFERENCES

1. J. M. Beckers, "Detailed compensation of atmospheric seeing using multiconjugate adaptive optics," in *Proc. SPIE Vol. 1114, p. 215-0, Active Telescope Systems, Francois J. Roddier; Ed.*, pp. 215–0, Sept. 1989.
2. J. L. Codona and R. Angel, "Imaging extrasolar planets by stellar halo suppression in separately corrected color bands," *ApJ Letters* **604**, pp. L117–L120, Apr. 2004.
3. J. P. Veran, F. Rigaut, H. Matre, and D. Rouan, "Estimation of the adaptive optics long-exposure point-spread function using control loop data," *JOSA A* **14**, p. 3057, 1997.
4. M. Le Louarn, "Multi-Conjugate Adaptive Optics: a PSF study," in *Beyond conventional adaptive optics : a conference devoted to the development of adaptive optics for extremely large telescopes. Proceedings of the Topical Meeting held May 7-10, 2001, Venice, Italy. Edited by E. Vernet, R. Ragazzoni, S. Esposito, and N. Hubin. Garching, Germany: European Southern Observatory, 2002 ESO Conference and Workshop Proceedings, Vol. 58, ISBN 3923524617, p. 217*, pp. 217–+, 2002.
5. B. L. Ellerbroek, "A wave optics propagation code for multi-conjugate adaptive optics," in *Beyond conventional adaptive optics : a conference devoted to the development of adaptive optics for extremely large telescopes. Proceedings of the Topical Meeting held May 7-10, 2001, Venice, Italy. Edited by E. Vernet, R. Ragazzoni, S. Esposito, and N. Hubin. Garching, Germany: European Southern Observatory, 2002 ESO Conference and Workshop Proceedings, Vol. 58, ISBN 3923524617, p. 239*, pp. 239–+, 2002.
6. B. Stroustrup, *The C++ Programming Language*, Addison Wesley, Boston, 1997.

7. R. N. Goldman, "Vector geometry: A coodinate-free approach," *ACM SIGGRAPH 1987 Course 19* , 1987.
8. J. W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, New York, second ed., 1996.
9. M. V. Papalexandris and D. R. Redding, "Calculation of diffractive effects on the average phase of an optical field," *JOSA* **17**, p. 1763, 2000.
10. J. Stoer and R. Burlisch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1993.
11. R. J. Sasiela, *Electromagnetic Wave Propagation in Turbulence*, Springer-Verlag, New York, 1994.
12. R. G. Lane, A. Glindemann, and J. C. Dainty, "Simulation of a komolgorov phase screen," *Waves in Random Media* **2**, p. 209, 1992.
13. E. M. Johansson and D. T. Gavel, "Simulation of stellar speckle imaging," in *Proc. SPIE Vol. 2200, p. 372-383, Amplitude and Intensity Spatial Interferometry II, James B. Breckinridge; Ed.*, pp. 372–383, June 1994.
14. B. L. Ellerbroek and F. J. Rigaut, "Scaling multiconjugate adaptive optics performance estimates to extremely large telescopes," in *Proc. SPIE Vol. 4007, p. 1088-1099, Adaptive Optical Systems Technology, Peter L. Wizinowich; Ed.*, pp. 1088–1099, July 2000.
15. J. W. Hardy, *Adaptive Optics for Astronomical Telescopes*, Oxford University Press, New York, 1998.
16. M. Troy and G. Chanan, "Diffraction effects from giant segmented-mirror telescopes," *Applied Optics* **42**, p. 3745, 2003.
17. B. R. Oppenheimer, "Direct detection of brown dwarf companions of nearby stars," *Ph.D. Thesis* , 1999.
18. J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, New York, 1994.
19. A. Alexandrescu, *Modern C++ Design*, Addison Wesley, Boston, 2001.
20. W. Pence, "CFITSIO, v2.0: A New Full-Featured Data Interface," in *ASP Conf. Ser. 172: Astronomical Data Analysis Software and Systems VIII*, pp. 487–+, 1999.
21. M. Frigo and S. G. Johnson, "Fftw: An adaptive software architecture for the fft," in *ICASSP conference proceedings v. 3*, pp. 1381–1384, 1998.
22. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, third ed., 1999.