Modelling and Simulation of Adaptive Optics in the Scilab/Scicos Environment

Chen jingyuan, Gan guangyong, Tao yingxue
The institude of Applied Physics and Computational Mathematics, Beijing, 100088, China
jingyuan_chen@yahoo.com.cn

Abstract: SciAO is an open source, cross-platform, and user-friendly toolbox based on the Scilab/Scicos environment for modeling and simulation of wave optics, especially the adaptive optics system. This paper describes the functionality available in SciAO, gives some application examples, and discusses future plans for this toolbox.

Keywords: model and simulation, adaptive optics, Scilab/Scicos, open source

1. Introduction

Through nearly 40-years development, Adaptive optics (AO) [1] is now a mature technology widely used in optical astronomy, high energy laser (HEL) systems and other modern optical equipments. At the same time, AO is also a dynamic field of research, new architectures, algorithms, and applications being proposed have pushing the original concept (first suggested by Babcock at 1953) to the technological limits (e.g. MCAO: Multi-Conjugate AO system).

To grasp the behavior and the performance analysis of these optical systems, a theoretical analyse is obviously the first choice, while this approach can’t work well in practise, partly because the overly complexity of AO systems and its complex working enviroment, partly because a number of problems are not open to analytical solution. In fact, this kind of study involves the complex nature of the turbulent atmospheric environment, and also the complexity of interactions of many connected instruments and concepts in a whole AO system. Within this background of the growing complexity of these system architectures and the expense of their implementation, time domain numerical simulations can play a very useful role in providing a quantitative evaluation of their capabilities. The advantage of a simulation lies in the fact that absolute truth is known, and the fact it is very cheap and controllable. 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 to evaluate 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 (Point Spread Function) 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.

There are a large number of softwares that can execute adaptive optics simulation. Most of them are commercial or proprietary products, and its inner mechanism is not known to other researchers. Some simulation softwares are open source, while these code are almost all written in FORTRAN, or C/C++, and they have not good interfaces for other people to use. Many of today's codes (e.g. AO tools developed by tOSC) are written entirely or partly in MATLAB, which has the advantages of an extensive library of math routines, very fast matrix operations, plenty of other toolboxes and very vell graphical outputs. The main limitations of MATLAB are the cost of the licenses and the slow speed for non-matrix iterative calculations.

Recently we have been developing a toolbox (we called it SciAO) based on Scilab/Scicos [2] environment in which we can model and simulate most adaptive optics devises or other AO-related problems. Although this toolbox is mainly be designed to satisfy the requirement of AO simulations, we also moderately consider the needs to some other optics simulation system (e.g. the simulation of general Wave optics and imaging devises) and designed some software modules for them, so this toolbox can also be used to simulating other optics problems beyond the AO field. Most of the programs of our SciAO toolbox are written by C/C++ language or based on some excellent open source C/C++ libraries, so it is efficient and cross-platform. At the same time, we also fully make use of the powerful graphical and interface program ability, especially the dynamic modeler and simulator (Scicos) of the Scilab environment, so it is very easy for other researchers or beginners to learn and use this toolbox to simulate their peculiar adaptive or other optical problems. Our toolbox will be released under open-sourced GPL license and all other people who are interested in this software can download, modify, or use it if they abide this license.

In this paper, we will give a general description of the SciAO toolbox (now the steady version is 0.2) and give some examples of simulation applications. At the last we also give some plans of the functional extension for SciAO. While first we need give a introduction of the basic concept of adaptive optics and its simulation, which is a unacquainted field to most of other researchers.

2. Adaptive optics concept and some relevant simulation software

Adaptive optics is a newly developed technology in order to improve the performance of the reflecting telescopes by reducing the effects of optical aberration or distortions, especially the distortions caused by the Earth's atmosphere. An classical adaptive optics system tries to correct these distortions, using quality-of-image detector (sensors), tip/tilt mirrors and deformable mirrors (correctors), and a controllable computer that receives input from the detector and calculates the optimal deformation of the mirror, and the corresponding control signals which will be used to make the deformable (or other corrector devises) change its shape or move. Figure 1 is an outline plot of the main components and its basic work diagram of a classical adaptive optics (SCAO: Single-Conjugate AO system).

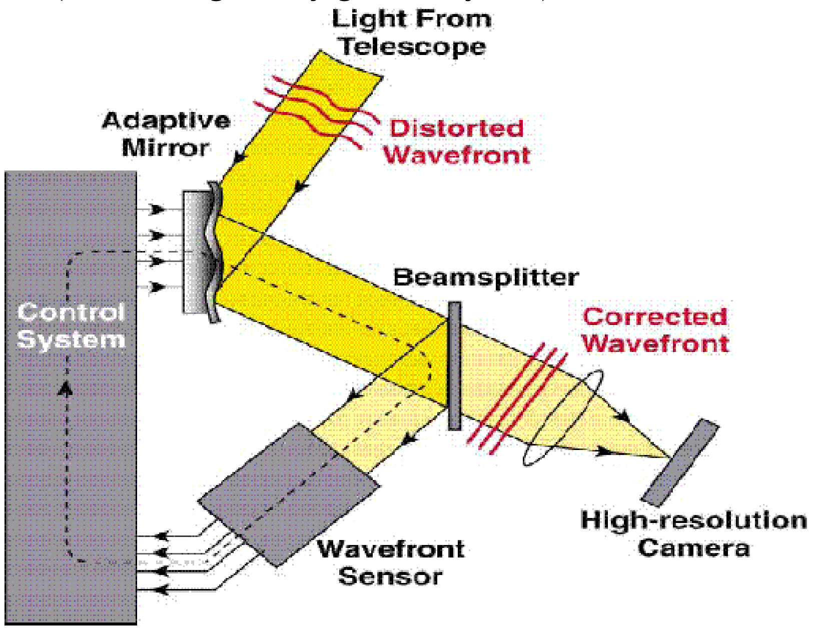


figure 1: the structure of a classic SCAO system

Generally say, AO simulation code can be roughly divided into two types. At the lower level are the physical and engineering codes that try to model and predict the complex interactions or performance based on first principles. They can predict detailed system performance characteristics such as evolution of wave field phase, intensity profiles, and a number of other information such as deformable mirror (DM) actuator commands or wavefront sensor pixel output, etc. These are also known as wave optics, time domain codes. At the top level are the scaling law or systems engineering code. These provide much faster predictions based on scaling laws or empirical fits to simulation or experimental results. At below, we will give a briefly description of some AO simulation codes which are relevant to our SciAO toolbox. They are WaveTrain, CAOS, LightPipes, and Arroyo. All of them belong to the physical or engineering codes.

WaveTrain [3] is a user-friendly wave and adaptive optics propagation code mainly developed and distributed by MZA Associates Corporation, USA, under funding from Air Force Research Laboratory, USA. It has been implemented using Tempus (another main product of MZA), which is a general purpose simulation development environment similar to the Simulink toolbox of Matlab. Using Tempus as the foundation, WaveTrain construct a lot of component library for wave and adaptive optics, where each type of software component models a different system component or physical effect.

CAOS [4] (Code for Adaptive Optics System) is a set of software tools specifically designed to allow the modeling and simulating of any kind of adaptive optics system originally developed in the framework of the “TMR Network on Laser guide star for 8 meter class telescopes” funded by the European Community. Similar to WaveTrain, CAOS is also a user-friendly AO simulation tool, and is essentially a set of modules designed to be used within a graphical programming environment – the CAOS Application builder (another clone of Simulink), where the data flow between modules can be defined, and the parameters of each module can be set. Dissimilar to WaveTrain, CAOS is an open source software, but it is implemented in the IDL (Interactive Data Language, designed by RSI Corp).

LightPipes [5] (written by Dr. Gleb Vdovin) is a tool that can model free space diffractive propagation and the effects of various optic devises such as apertures and lenses. The simulation used the pipe command to transfer the wavefront data between individual programs. This tool contains no support for modeling atmospheric turbulence and other adaptive optical devises. Most of this programs are written in C and is open-sourced. It has also a Matlab version, but this version is only can run under limited mode (only 64x64 grid dimension at most).

Arroyo [6], developed by Dr. Matthew Britton at California Institute of Technology, is an open source C++ class library designed for modeling and simulation of electromagnetic wave propagation through atmospheric turbulence and adaptive optics system. Based on the object-oriented programming methodology and other modern software engineering technology (e.g. coordinate free geometric programming, the object factory design pattern, etc), this library is an excellent software project which achieves powerful abstractions and good modularity, reusability, and extensibility. This library also supports parallelized computation.

3. SciAO: goals and schemes

WaveTrain and CAOS are user-friendly, but they are proprietary in essential (Though SCAO is open source, the IDL is proprietary), and their efficiencies are not very well, because they are implemented by higher language. LightPipes and Arroyo are more efficient, while it is hard to use them to simulate particular optics problems.

We want to develop a user-friendly simulation tool which can model and simulate wave or adaptive optics easily, at the same time, it must be as efficient and credible as other tested FORTRAN or C/C++ code. We also expect it is open-sourced and all of other people in or out the filed of adaptive optics can use and improve it arbitrarily.

We select the Scilab as our fundamental developing platform (so it called), because Scilab is open-sourced, and easy to learn and use for beginner. More important, Scilab offers interfaces to external libraries written in other lower level languages such as C and FORTRAN, which make it possible to use legacy code and other open-source software. Furthermore, Scilab has a powerful visual modelling and simulation enviornment, called Scicos, which is very semblable to the Simulink of Matlab, or the Tempus of WaveTrain, or the Aplication Builder of CAOS. Using Scicos, a simulation can be built by connecting together the required occurrences of the desired modules (blocks), respecting only the logical constraints given by their formalized input and output type, and then the block diagram will be analysed and executed by Scicos itself. Complex simulation application are thus simply created by assembling the elementary building blocks in a straightforward manner, so that the user can concentrate on the scientific aspects of his special problems, while mundane coding problems are managed by some automatic mechanism.

In developing this tool we did not "reinvent the wheels". All of the basic techniques and algorithms used in wave optics which had already been implemented and tested for previous wave optics codes, and all of the open-source code which we can acquire, can be used by our toolbox. Where it made sense, we used the legacy or open-source code directly. In some cases minor modifications were needed to satisfy integration requirements, and in some cases it was simpler to rewrite the code. For open-source codes which we have used in SciAO, we refer to Arroyo and LightPipes.

4. Status and functional description of SciAO (version 0.2)

We have worked on this toolbox almost a year and now it has been able to model and simulate some simple wave and adaptive optics problems. In this part, we will describe the basic structure and function of SciAO.

SciAO contains several elements, the main are:

- Some external optics routines libraries. They are kernels of the SciAO, and all of the optics computation are actuated by them. Now two external library has been integrated into SciAO: lightPipes and arroyo. We have made some modification to their original codes in order to let them accord with Scilab/Scicos environment, or let them be able to cross-platform run under Microsoft Windows/Visual Studio .net 2003 and Gnu/Linux. Optics libraries also include another routines library we called Wrapper, which is some C wrappers to Arroyo C++ class library. We need such a C wrapper library because Scilab/Scicos now has not been able to interface C++ class library directly.
- C routines libraries that interface external optics library to Scilab/Scicos. There are two such C routines libraries: the interface function library of Scilab functions for Optics, and the computational function library of the Scicos blocks for Optics. Now we have constructed about one hundred Scilab functions to optics, and about thirty Scicos blocks to optics.
- Some Scilab scripts, include the interface functions of Scicos blocks for Optics, and some other useful functions written by original or SciAO' Scilab functions. There are also some TCL/TK scripts we use them to construct some beautiful GUI.
- Many examples of Scicos diagrams and Scilab simulation scripts. We have written these examples using the Scilab functions and Scicos blocks of our SciAO toolbox, so that beginners can learn and grasp the use of our toolbox. Further, users can construct their special simulation models of wave optics or adaptive optics at the base of these examples we have provided.

At below, we will give an outline description of the main Scilab functions and Scicos blocks of our SciAO toolbox. We start from the blocks. Figure 2 gives out the Scicos palette of SciAO.

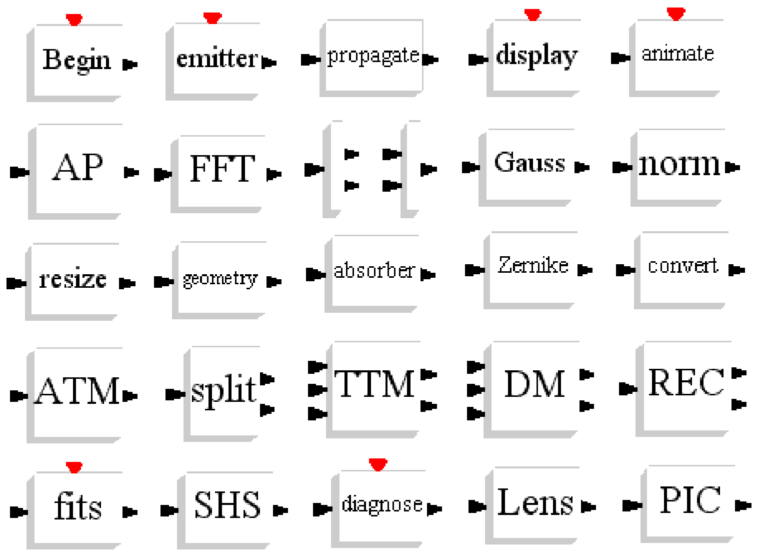


figure 2: the optical Scicos palette in SciAO

As Figure 2 points out, there are now more than twenty optical Scicos blocks in SciAO. In them, the blocks labeled Begin and Emitter can be used to model optical “sources”, while Display, Animate, Fits, and Diagnose blocks model optical “sink” (or say it detectors). All the other optical blocks in this Figure are used to model and simulate various optical equipment, such as atmospheric refractive layers (ATM), deformable mirrors (DM), Shack-Hartmann sensors (SHS), apertures (AP), beam splitter (Split), lens (Lens), reconstructor of wavefronts (REC), proportional integral controller (PIC), or other optical transforms, such as geometrical optics transform (the Geometry block), the normalization of intensity (Norm), and frame transforms (Convert), fast Fourier transform (FFT), and so on. We will add more and more blocks to let it be able to model other optical devises or transform at later versions of SciAO.

Besides above Scicos blocks for optics, we also write a number of Scilab function to model various optical equipments or optical processes and now the number of these functions is about one hundred. Now we give them some description. They can be include in these types (we list most of them, and give some brief description for its function in brackets):

- Emitters: begin (form unit plane wave); emitter (construct light emitter); gauss (form Gauss beam); pl_wave (form general plan wave); sp_wave (form spherical wave or point source); etc.
- Detector or Output: dwf_fits (write wavefront to fits file); cros_out (write the cross sections of a field); file_pgm (write field to a pgm files); file_ps (write field to a ps file); file_int (write the intensity of field to a file); file_pha (write the phase of field to a file); znk_fits (write Zernike mode to a fits file); ref_atm_lay_fits (write a refractive atmospheric layer to a fits file); atm_mod_layers_fits (write all the layers of a atmospheric model to fits files); reconstructor_fits (write the reconstructor of a AO system to a fits file); etc.
- Wavefronts and Propagators: wavefront_header (create the wavefront

header of a field); set_dwfh_timestamp (set the timestamp of a wavefront); set_dwfh_curvature (set the curvature of a wavefront); fresnel (Fresnel propagator); forvard (FFT or spectra method propagator); forward (Direct integration method propagator); lens_forvard (FFT or spectra method propagator for spherical wave); lens_fresnel (Fresnel propagator for spherical wave); steps (Finite difference method propagator); geom_propagation (geometric optics propagator); near_angular (near field angular propagator); near_fresnel (near field Fresnel propagator); far_fresnel (far field Fresnel propagator); far_fraunhoffer (far field Fraunhoffer propagator); fresnel_grt (far field Fresnel and Goertzel-Reinsch tranform propagator); fraunhoffer_grt (far field Fraunhoffer and Goertzel-Reinsch tranform propagator); etc.

- Telescope aperture and its transforms: circ_ap (circular aperture); rect_ap (rectangular aperture); hex_ap (hexagonal aperture); annul_ap (annular aperture); spid_annul_ap (spidered annular aperture); til_hex_ap (tilted hexagonal aperture) ; aperture_transform (transform a wavefront by aperture);
- Atmosphere models: power_law (create power law of turbulent spectrum); von_karman_power (Von-Karmman power spectrum); greenwood_power (Greenwood power spectrum); null_inner (null inner scale of a power law); exponent_inner (Tartaskii exponent inner scale); frehlich_inner (Frehlich exponent inner scale); power_spectrum (isotropic power law spectrum of turbulence); null_subm (null subharmonic method); quad_pix_subm (quad-pixel subharmonic method); lane_subm (Lane subharmonic method); johansson_gavel_subm (Johansson-Gavel subharmonic method); general_subm (generalized subharmonic method); ref_atm_lay (create refractive atmospheric layer); ref_atm_lay_transform (transform a wavefront by refractive atmospheric layer); hardy_wind (create Hardy wind); get_hardy_wind_vectors (get the wind vectors of Hardy wind model); ref_atm_model (create generalized refractive atmospheric model); ell_cer_pac_mod (Ellerbroek Cerro Pachon atmospheric model); ell_mau_kea_mod (Ellerbroek Mauna Kea atmospheric model); pal_dimm_mass_mod (Palomar DIMM MASS atmospheric model); huf_val_mod (Hufnagel Valley atmospheric model); slcsat_day_mod (SLCSAT daytime atmospheric model); slcsat_night_mod (SLCSAT nighttime atmospheric model); tmt_srd_v13_cn2_mod (TMT SRD v13 Cn2 atmospheric model); gemini_glao_study_mod (the Gemini GLAO study model); get_atm_mod_dwfh (get the wavefront of an atmospheric model); get_atm_mod_layers (get all the refractive layers of an atmospheric model); atm_mod_lay_number (get the number of refractive layers of an atmospheric model); atm_mod_lay_heights (get the heights vectors of a model); etc.
- Optical devises and transform: lens (create a lens); file_ter (general file filter); zernike (Zernike mode filter); random (random filter); b_mix (beam mixer); b_split (beam splitter); l_amplify (Laser amplifiers); tilt (tilt modes of the field); absorber (beam absorber or amplifier); etc.
- AO appropriative devises and its transforms: lnslt_array (Shack Hartmann

lenslet array); lnslt_arr_transform (transform wavefront by Shack Hartmann Sensor); create_shcentroids (create Shack-Hartmann centroids); actuator_array (create actuator array for a deformable mirror); ideal_dm (create ideal deformable mirror); ideal_dm_transform (transform wavefront by a ideal deformable mirror); set_dm_timestamp (set the timestamp of a deformable mirror); set_dm_actuator_positions (set the positions of the actuators for deformable mirror); set_dm_actuator_commands (set the command vectors of the actuators for deformable mirror); ideal_ttm (create ideal tip-tilt mirror); set_ttm_timestamp (set the tamestamp of a tip-tilt mirror); set_ttm_commands_vector (set the command vectors of a tip-tilt mirror); ideal_ttm_transform (transform a wavefront by a tip-tilt mirror); pi_controller (create a general proportional integral controller); ttm_pi_controller (create proportional integral controller for a tip-tilt mirror); dm_pi_controller (create proportional integral controller for a deformable mirror); ttm_pi_update (update the proportional integral controller of a tip-tilt mirror); dm_pi_update (update the proportional integral controller of a deformale mirror); create_ttm_commands (create the command vector of a tip-tilt mirror); create_dm_commands(create the command vector of a deformable mirror); reconstructor (create a wavefront reconstructor for a AO system); zernike_residuals (get Zernike mode residual of the corrected wavefront); zonal_residuals (get zonal residual of the corrected wavefront); atm_mod_lays_transform (transform wavefront by an atmospheric model); etc.

- Others: znk_mod (create Zernike polynomial); set_znk_cos_coef (set the cosine coefficient of a Zernike polynomial); set_znk_sin_coef (set the sine coefficient of a Zernike polynomial); pixel_array (create a 2D pixel array); set_pix_arr_data (set the data of a 2D pixel array); three_point (create a point in 3D space); three_vector(create a vector in 3D space); three_frame create the frame in 3D space); three_translation (create a translation in 3D space); three_rotation (create a rotation in 3D space); convert (convert the frame between plane and spherical frame); normal (normalize the wavefront to unit intensity); pip_fft (FFT spactial filter); interpol (interpolate a field); strehl (get the Strehl ratio of a field); etc.

For more information about the Scilab functions or Scicos blocks which we has constructed for our SciAO toolbox, please consult the online help documents.

5. Application examples

In this part, we will give two examples to demonstrate how to model and simulate optical problems using the SciAO toolbox. The first example is about Young’s interferogram, while the second relates to SCAO systems. For more examples, consult the “examples” directory in our SciAO release version.

- Young’s interferogram

Consider such an experiment: suppose a completely coherent plane light, having unit amplitude and 550 nm wavelength , vertically irradiate to a board possessing two circlular holes, apart from 0.5 mm and both have radius R=0.25mm, we want to oberserve the interferogram at the distance 30 cm behind the plane.

We can easily model such an experiment using the Scilab functions we have designed for SciAO. Below is a Scilab script which simulate such a physical process:

```
m=1; cm=1e-2*m; mm=1e-3*m; nm=1e-9*m;
ngrid=256; size=5*mm; lamda=550*nm;
radius=0.25*mm; shift=0.5*mm; distance=30*cm;
init_field = begin(size,lamda,ngrid);
circ_ap_1 = circ_ap(init_field,radius,shift);
circ_ap_2 = circ_ap(init_field,radius,-shift);
mix_field = b_mix(circ_ap_1,circ_ap_2);
dwf_fits(mix_field,"plane");
interf_field = forvard(mix_field,distance);
dwf_fits(interf_field, "interferogram");
intensity=field_int(interf_field);
x=1:ngrid; y=1:ngrid; grid=[1,1,ngrid,ngrid];
grayplot(x,y,intensity,strf="030",rect=grid);
```

In above script, we have written field to two FITS files (Flexible Image Transport System, it is the standard data format used in astronomy. Of cause, you can write them into other format, such as PS or PGM, etc., and we have also provided some other functions to support these format in SciAO): plane.fits and interferogram.fits, and at the same time plotted the interferogram directly.

A more intuitional method to model and simulate in SciAO is using the optical Scicos blocks which we have designed for SciAO. Figure 3 gives out a diagram that can model above Young's experiment. In this diagram, we have packed the board to a super-block for concision, whose content is revealed in Figure 4.

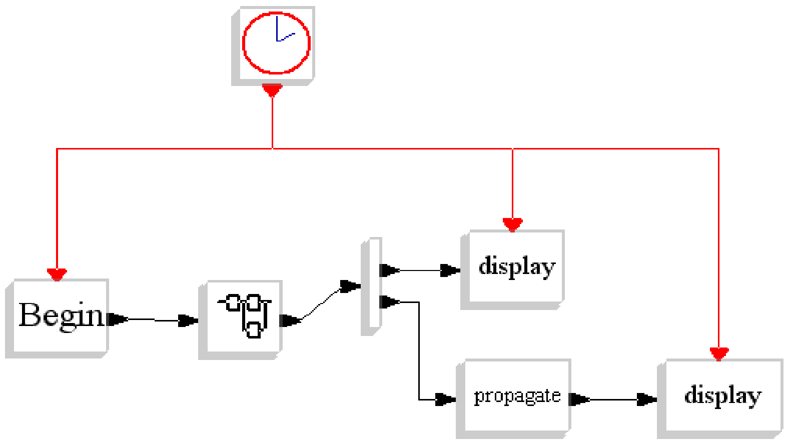


figure 3: the diagram to model Young's interfere

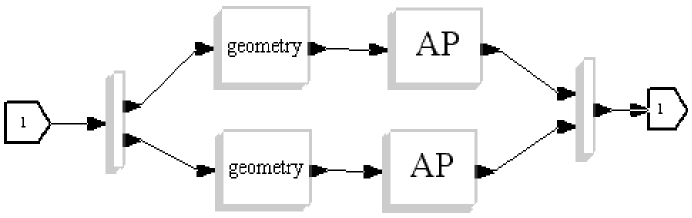


figure 4: the contents of the super-block in figure 3

Two methods (using Scilab functions or using Scicos blocks of SciAO) is equivalence. We give out the results of simulation in figure 5:

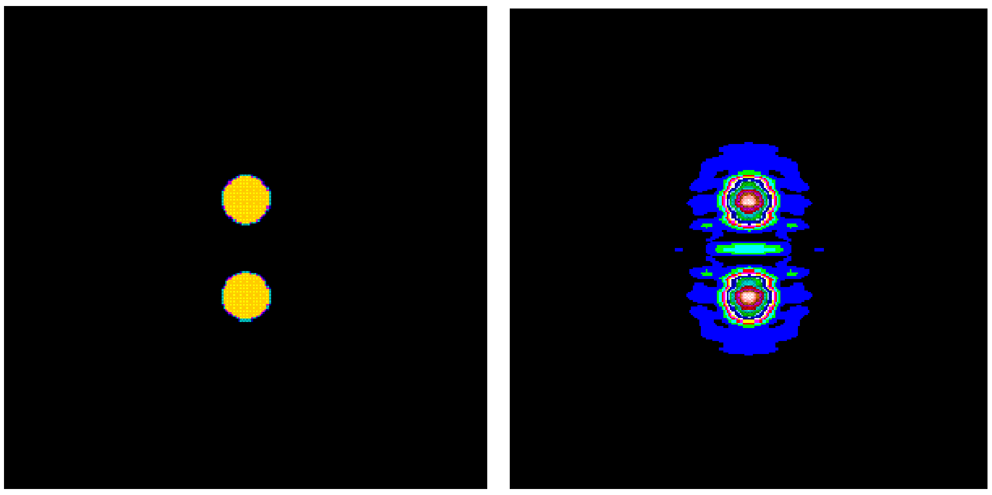


figure 5: left is the field at the behind of the board
right is the interferogram

- SCAO system

Analogously, we can model and simulate AO systems by two equivalent methods in SciAO. We first give a diagram example in figure 6 to simulate a simple SCAO system.

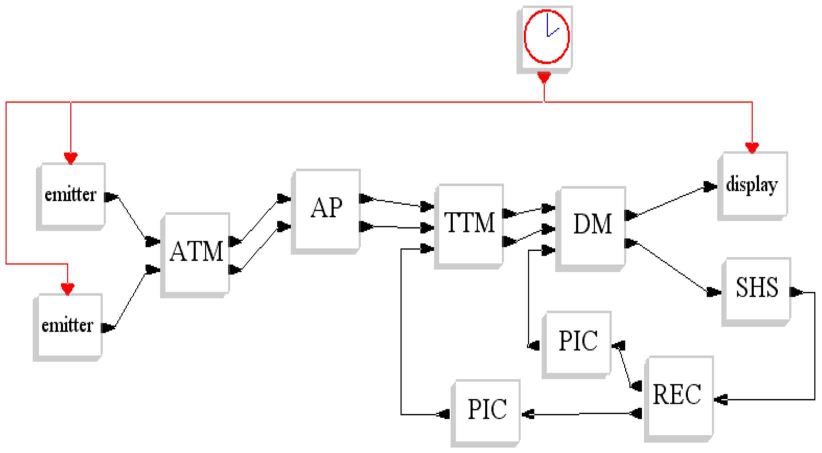
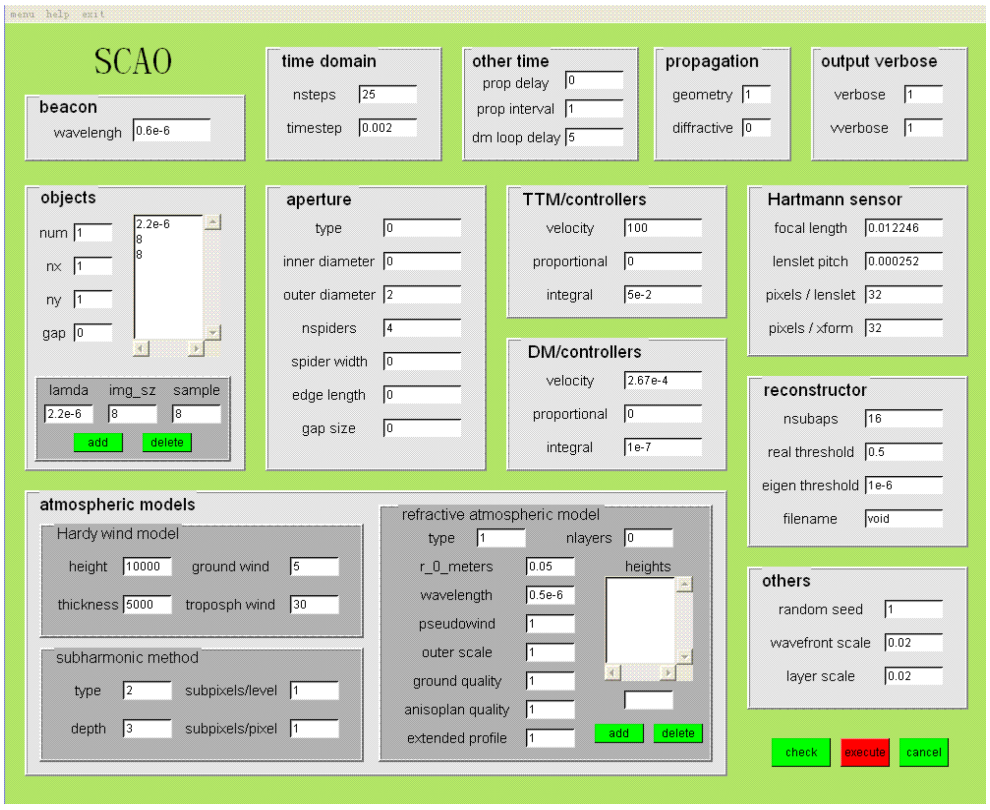


figure 6: the diagram to model a SCAO system

The Scilab scripts that model this simple SCAO system is lengthy, so we do not give them out here. All the parameters to simulate this system are about 70, and we have designed a GUI to simplify the input process. The GUI is plot at Figure 7:



beacon

wavelengh	0.6e-6

time domain

nsteps	25
timestep	0.002

other time

prop delay	0
prop interval	1
dm loop delay	5

propagation

geometry	1
diffractive	0

output verbose

verbose	1
vverbose	1

objects

num	1
nx	1
ny	1
gap	0

2.2e-6
8
8

lamda	img_sz	sample
2.2e-6	8	8

aperture

type	0
inner diameter	0
outer diameter	2
nspiders	4
spider width	0
edge length	0
gap size	0

TTM/controllers

velocity	100
proportional	0
integral	5e-2

DM/controllers

velocity	2.67e-4
proportional	0
integral	1e-7

Hartmann sensor

focal length	0.012246
lenslet pitch	0.000252
pixels / lenslet	32
pixels / xform	32

reconstructor

nsubaps	16
real threshold	0.5
eigen threshold	1e-6
filename	void

atmospheric models

Hardy wind model

height	10000	ground wind	5
thickness	5000	troposph wind	30

subharmonic method

type	2	subpixels/level	1
depth	3	subpixels/pixel	1

refractive atmospheric model

type	1	nlayers	0
r_0_meters	0.05	heights	
wavelength	0.5e-6		
pseudowind	1		
outer scale	1		
ground quality	1		
anisoplan quality	1		
extended profile	1		

others

random seed	1
wavefront scale	0.02
layer scale	0.02

figure 7: the GUI to model a SCAO system

In Figure 8 we give out two figures to demonstrate the process of adaptive correction:

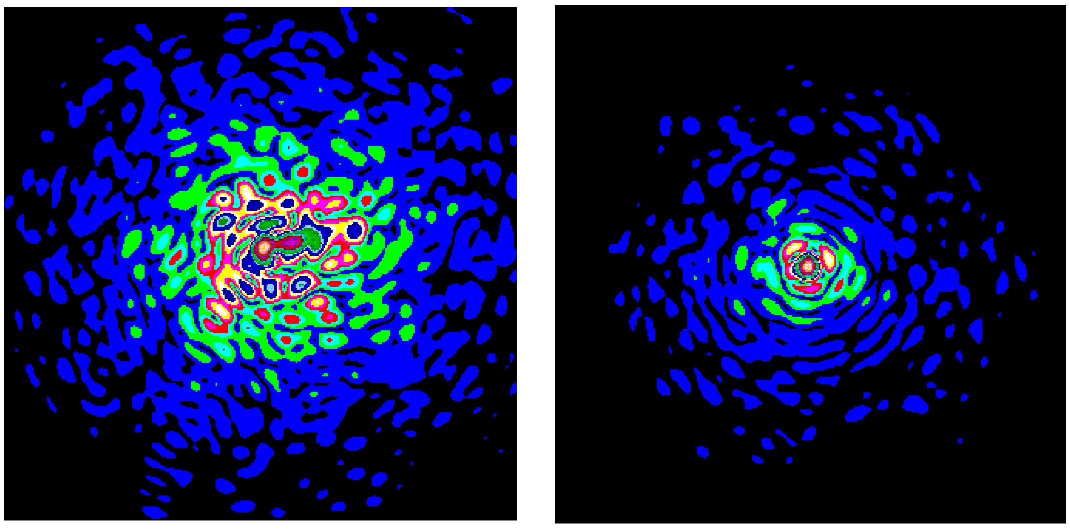


figure 8: the left is an uncorrected image and the right a corrected image

6. Conclusion, future extensions and plans

The objective that we select the Scilab/Scicos environment and design SciAO toolbox for it is to make an optical simulation much more easier and user-friendly than original FORTRAN, or C/C++ codes, without sacrificing fidelity or efficiency. We also expect other researchers in the field of modeling and simulation of adaptive optics to use, correct, and improve this toolbox, so we release it under GPL license. We believe we have met our original objective. Users now can build models from scratch, or simply copy an existing model, then modify it to suit their needs. More advanced users can create their own components at the top of our fundamental C/C++ rountines or our optical Scilab functions, which can then be used and reused just like those in our Optics library or other Scilab library.

At the same time, there remains considerable room for improvement. Now SciAO is only at its infancy and is very much under development. To model and simulate more complex application problems in wave or adaptive optics, we need construct other software modules. Other than, the documentation is never up to date; The component library is far from complete, and many of component interfaces could be improved; The GUI is getting better and better, but there is always more to do; And so on. We are hard at work on all these things.

7. References

[1] Hardy, Adaptive Optics for Astronomical Telescopes, Oxford Press, 1998

[2] http://www.scilab.org/ and http://www.scicos.org/

[3] http://www.mza.com/

[4] http://www.arcetri.astro.it/caos/

[5] http://www.okotech.com/software/lightpipes/

[6] http://eraserhead.caltech.edu/