CAOS – a numerical simulation tool for astronomical adaptive optics (and beyond)

Marcel Carbillet[a⋆], Christophe Vérinaud[b], Mario Guarracino[c], Luca Fini[a], Olivier Lardière[a], Brice Le Roux[a], Alfio Puglisi[a], Bruno Femenía[d], Armando Riccardi[a], Barbara Anconelli[e], Serge Correia[f], Mario Bertero[e], Patrizia Boccacci[e]

[a] INAF–Osservatorio Astrofisico di Arcetri (OAA), largo E. Fermi 5, I-50125 Firenze
[b] European Southern Observatory, Karl-Swarzschild str. 2, D-85748 Garching-bei-München
[c] CNR–High Perf. Computing and Networking Inst., via P. Castellino 111, I-80131 Napoli
[d] GTC Project, Instituto de Astrofísica de Canarias, C/ vía Láctea s/n, E-38200 La Laguna
[e] INFM and DISI, Università di Genova, Via Dodecaneso 35, I-16146 Genova
[f] Astrophysikalisches Institut Potsdam, An der Sternwarte 16, D-14482 Potsdam

ABSTRACT

We present recent developments of the CAOS "system", an IDL-based Problem Solving Environment (PSE) whose original aim was to define and simulate as realistically as possible the behavior of a generic adaptive optics (AO) system, from the atmospheric propagation of light, to the sensing of the wave-front aberrations and the correction through a deformable mirror. The different developments made through the last 7 years result in a very versatile numerical tool complete of a global graphical interface (the CAOS Application Builder), and different specialized scientific packages: the original one designed for AO system simulations (the Software Package CAOS), an image reconstruction package with interferometric capabilities (the Software Package AIRY), and a more recent one being built and dedicated to multiconjugate AO (the Software Package MAOS). We present the status of the whole CAOS "system"/PSE, together with the most recent developments, including parallelization strategy considerations, examples of application, and plans for the next future.

Keywords: adaptive optics, post-AO imaging, interferometric imaging, numerical simulations, parallelization

1. INTRODUCTION

Due to the general complexity of adaptive optics (AO) systems, performance analysis implies a number of problems to be solved that hardly have an analytical solution. In fact this kind of study involves the complex nature of the turbulent atmosphere itself, and also the fact that a number of competitive instrumental concepts are often to be deeply analyzed and compared (e.g. pyramid wave-front sensors vs. Shack-Hartmann ones, piezo-stacked mirrors vs. adaptive secondaries, different reconstruction strategies, etc.). As a consequence, numerical simulations, not always end-to-end but often extremely detailed at least for a particular physical process or piece of hardware behavior to be studied, are necessary. Moreover, a number of novel AO concepts are also being studied for facing new on-the-edge applications, such as extrasolar planet searching (e.g for the VLT-Planet Finder), wide-field high-angular-resolution astronomy (the ESO multiconjugate (MC) AO demonstrator MAD for VLT, the interferometric MCAO system LINC-NIRVANA for LBT, the Gemini laser-based system, etc.), or the development of the different projects of Extremely Large Telescopes (ELTs), pushing the limits of the original concept.

Studies concerning the coupling of AO systems with the subsequent instrumentation (interferometry, coronagraphy, spectroscopy, etc.), are also of central importance, and, in the same area of concerns, image reconstruction algorithms, especially when considering post-AO images, have also an important role to play within the

⋆ On move from OAA to Laboratoire Universitaire d'Astrophysique de Nice, UMR 6525, Université de Nice–Sophia Antipolis, Parc Valrose, 06108 Nice Cedex 02, France – Correspondance e-mail: marcel.carbillet@unice.fr

whole ensemble of techniques/tools that have to be developed in order to better benefit from the performance of the instrumentation, optimizing so the scientific return.

In this communication we present the CAOS "system" (where CAOS stands for Code for Adaptive Optics Systems), a Problem Solving Environment (PSE) which original aim was to define and simulate as realistically as possible the behavior of a generic AO system, from the atmospheric propagation of light, to the sensing of the wave-front aberrations and the correction through a deformable mirror. The different developments made through the last 7 years result in a very versatile numerical tool complete of a global graphical interface (the CAOS Application Builder), and different specialized scientific packages: the original one designed for AO system simulations (the Software Package CAOS), which integrates state-of-the-art physical modeling and is already distributed among a large community of AO-concerned astronomers/researchers, an image reconstruction package with interferometric capabilities (the Software Package AIRY – that stands for Astronomical Image Restoration with interferometrY), distributed also among its own community of users, and a more recent package being built and dedicated to multiconjugate AO (the Software Package MAOS – that stands for Multiple-reference multiconjugate Adaptive Optics Simulations). We present the status of the whole CAOS "system"/PSE, together with the most recent developments, including parallelization strategy considerations, examples of application, and plans for the next future.

Therefore, the paper is organized as follows. In Sect. 2 we rapidly describe the global structure and main features of our programming environment. In Sect. 3 we give the status of the Software Package CAOS, highlighting the more recent modeling features introduced. Section 4 is then dedicated to a brief description of the Software Package AIRY, together with an example of application when coupling it with the Software Package CAOS. The Software Package MAOS is also presented within this section. In Sec. 5 we then describe our work for parallelizing CAOS, and our concluding remarks are given in Sect. 6.

2. THE CAOS APPLICATION BUILDER

The structure under the CAOS "system" is modular. This means that each elementary physical process of a given simulation is modeled within a specific module – like, in the AO case, the turbulence in each atmospheric layer, the propagation of light from a source to the observing telescope and through the turbulent layers, the wave-front sensing, the wave-front reconstruction, the time-filtering of the resulting deformable mirror commands, the wave-front correction, etc. Taking advantage from the CAOS Application Builder, a simulation can be built putting and connecting together the required occurrences of the desired modules, respecting the only logical constraint given by their formalized type of input/output. Complex simulation applications are thus simply created by assembling the elementary building blocks (representing the modules) in a straightforward manner, so that the user can concentrate on the scientific aspects of her/his problem, while mundane coding problems are managed by some automatic tools.

Each module comes with an individual Graphical User Interface (GUI) in order to set its own physical and numerical parameters, during the design step or independently in a later moment. In practice, each module is so defined by a standard group of function calls, a collection of parameters, and a typed definition of input(s) and output(s). It can support up to two inputs and two outputs, and it is represented within the *Application Builder* as a rectangular box with colored input handles (on its left side), and output ones (on its right side). Each color describes one of the pre-defined type of input/output: wave-front, image, commands, etc. Each *Software Package* also contains a library of utilities, as well as a detailed hyper-text help which can be called from each individual module GUI, and a set of examples of typical simulation projects that can be used as a starting point for new applications.

After the simulation design step is completed, the block diagram is analyzed by the *Application Builder* and the IDL code implementing the simulation program is generated. It can possibly be modified "by hand" in order to be completed with some additional task not provided by strictly using the existing modules. The whole structure of the simulation can be saved as a project that can be restored for later modifications and/or parameters upgrading. Beside the main simulation project one or more calibration project(s) might be previously designed and ran.

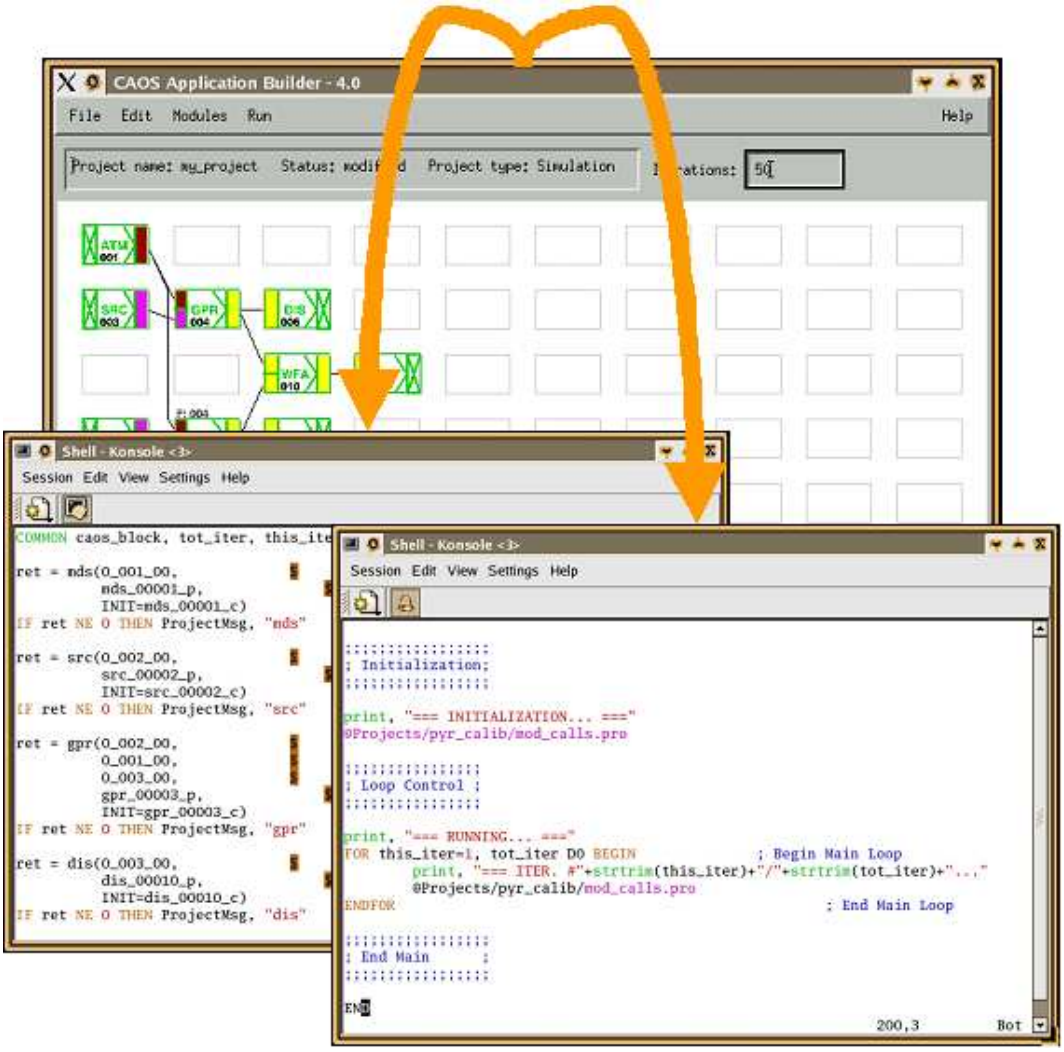

Figure 1. Background: the CAOS Application Builder showing a typical simulation design using here the modules of the Software Package CAOS. Foreground: the two routines that are automatically generated at the end of the simulation design phase.

Figure 1 shows the CAOS Application Builder, together with the automatically generated code: two IDL-written routines, one for the calls to the various modules required, and the other one for the general administration of the simulation project (graphical representation and global parameters). As already mentioned, this code is generated at the end of the design phase of the simulation, phase during which each required module is picked up from the "Module" menu in which all the modules of all the *Software Packages* installed are present, and then put in one of the pre-defined boxes within the *Application Builder* itself. The occurrence of a module is hence represented by a box with the name of the module (e.g. the turbulent atmosphere module is represented by the box named `ATM` in the figure) and with pre-defined inputs and outputs.

By clicking on the occurrence of a module (e.g. the occurrence of module `ATM` we were evoking before is number `001`), a GUI is opened in order to chose the various physical and numerical parameters related to the module itself (and only it). Global parameters (common to all modules) are limited to a strict minimum: only the total number of iterations and the current iterate number, and only if necessary. It is clear from Fig. 1 how

additional code can be easily implemented directly within the two automatically generated routines. Moreover, new modules can also very easily be implemented thanks to the template module included with the CAOS Application Builder distribution. As a consequence, it has to be noted that new *Software Packages* (dedicated to a given thematics not treated by the existing *Software Packages*) are also easy to build up. Let's also note that modules of different *Software Packages* can work together in a unique project, given the input/output structure type compatibility.

Everything, from the *Application Builder* (that now reached version 4.1) to each of the *Software Packages*, is implemented in IDL language, but efforts could be made in the next future to port the whole code to the newly developed GDL language, supposed to soon reach a complete equivalence to IDL version 6.0 (see `http://sourceforge.net/projects/gnudatalanguage/`).

3. THE SOFTWARE PACKAGE CAOS

Present version of the Software Package CAOS is 5.0, and it is the result of an almost 7-years-long work involving a considerable effort in the modeling of a number of physical processes involved, and in the numerical writing and distribution of this tool capable of simulating a wide range of astronomical optics situations, including atmospheric optics, AO, imaging, Fizeau interferometry, and more recently coronagraphy. Table 1 shows a complete list, together with a very brief description, of the modules present in the Software Package CAOS. A more detailed description of the whole set of modules can be found in Carbillet et al.[10]

Beside the Software Package CAOS, some other relevant AO numerical tools have been proposed, though not always as widely distributed and developed as CAOS, and often limited to a given type of AO. Among them we can cite: PAOLA, an IDL-written semi-analytical code dedicated to extremely large apertures AO[19]; LOST, IDL-based also and dedicated to layer-oriented MCAO simulations[4]; and the ESO parallel `C` codes, dedicated to extremely large apertures MCAO.[23] A number of cross-checks has been made, showing a global well agreement between the modules of the Software Package CAOS and these other codes, at least within a given set of relevant physical assumptions.

In the following, we first enumerate the main current applications that are being investigated thanks to our package, and then briefly describe its more recent developments.

3.1. Main Current Applications

A number of studies implying the Software Package CAOS in its successive versions has already been performed. Among them we can cite: a first performance evaluation performed for the first-light AO system of LBT,[7] the studies about the impact of partial AO-correction on the interferometric imaging capabilities of LBT,[5,8] the optimization of the deformable mirrors conjugation altitude in MCAO[15] and others studies in the framework of the GranTeCan AO system, the proved possibility to sense differential piston by using a pyramid sensor,[30] and the simulation studies concerning the ESO MCAO demonstrator MAD.[6,31] Moreover, and for a large part within these proceedings, a number of other studies have been, or are being, performed at least in part thanks to our package, for example for what concern the advanced performance analysis of the first-light AO system of LBT (WLBT) again,[9] some extreme AO studies made in the framework of the CHEOPS project for VLT-PF[20,25] or for Shack-Hartmann/pyramid comparison purpose,[33] a study for the AO system of the 20 m aperture telescope project TMT,[28] a characterization of the Lick 3 m AO system,[29] etc.

3.2. Recent Developments

The most recent developments made within the Software Package CAOS concern both the addition of new modules and the addition of new features in existing modules. As concerns new modules, and as it can be seen from Table 1, we have recently added a couple of coronagraphic modules, present in version 5.0 of the package. While for what concerns the more recent new features being developed, we briefly present (1) a modification of the wave-front sensor module `PYR` that can be generalized to any other module for which a low-light level (LLL) CCD can be used, (2) a series of upgrades of the reconstruction module `REC` permitting to consider more interesting reconstruction/command control strategies, and (3) a modification of the beam combiner module

Table 1. Descriptive list of the present modules of the Software Package CAOS (version 5.0).

Module	Purpose
Wave-front perturbation and image formation	
ATM - ATMosphere building	-to build the turbulent atmosphere (see also utility PSG - Phase Screen Generation)
SRC - SouRCe definition	-to characterize the guide star/observed object
GPR - Geometrical PRopagator	-to geometrically propagate the light
IMG - IMaGing device	-to make an image of the observed object
Wave-front sensing	
PYR - PYRamid wave-front sensor	-to simulate the pyramid sensor behavior
SLO - SLOpe calculator	-to compute the slopes from the pyramid signals
SH2/SHW - Shack-Hartman sensor modules	-to simulate the Shack-Hartmann sensor behavior
CEN/BQC - Centroiding modules	-to compute the slopes from the SH spots centroiding
Wave-front reconstruction and correction	
REC - wave-front REConstruction	-to reconstruct the wave-front
TFL - Time-FiLtering	-to apply time-filtering during reconstruction
DMI - Deformable MIrror	-to correct from the wave-front perturbations
Tip-tilt-specialized modules	
TCE - Tip-tilt CEntroiding	-to reconstruct the tip-tilt
TTM - Tip-Tilt Mirror	-to correct from the tip-tilt
LGS-oriented modules	
LAS - LASer characterization	-to define the laser characteristics
NLS - Na-Layer Spot definition	-to characterize the Sodium-layer behavior
Calibration-oriented modules	
CFB - Calibration FiBer characterization	-to define the fiber used for calibration purpose
CSQ - Command SeQuencer module	-to generate a sequence of commands
MCA - Make CAlibration module	-to make and save the calibration matrix
MDS - Mirror Deformation Sequencer	-to generate a sequence of mirror deformations
SCD - Save Calibration Data	-to save the calibration data
Other modelization modules	
IBC - Interferometric Beam Combiner	-to combine the light from two pupils
COR - CORonagraphic module	-to simulate various coronagraphic concepts
AIC - Achromatic Interferential Coronagraph	-to simulate the AIC concept
BSP - Beam SPlitter	-to split the light beam
Additional utilities	
WFA - Wave-Front Adding module	-to add or combine together wave-fronts
ATA - ATmosphere Adding module	-to add or combine together atmospheres
IMA - IMAge adding module	-to add or combine together images
STF - STructure Function calculator	-to compute the structure function from wave-fronts
SAV - SAVe structure	-to save any input/output structure (XDR format) (see also utility RST - ReSTore structure)
DIS - generic DISplay	-to display any input/output

IBC that permit to extend this interferometric module for pupil densification. These three last modifications will be available on a next future release following version 5.0.

Coronagraphy

Two coronagraphic modules have been added to the Software Package CAOS: module COR for "classical" coronagraphy (Lyot, Roddier & Roddier, four quadrant phase mask) for which the coding is very similar (basically three FFTs in a line of code), and module AIC for the Achromatic Interfero-Coronagraph concept. The implementation details were already discussed elsewhere[11] with an example of application implying the WLBT system in high-Strehl mode (K band, 30×30 sub-apertures, bright AO guide star, all the 672 modes of the LBT 672 adaptive secondary mirror applied) together with different coronagraphic concepts evoked before, and a double star with a faint companion.

Wave-front sensing/image formation: LLLCCD behavior implementation

Another type of developments concerns the consideration of CCDs of a new type : the LLLCCDs, which have the capability of giving a high internal gain, attaining so sub-electron RON level. As concerns wave-front sensing, this would permit hence to push the limiting magnitude. But, because of the essentially stochastic nature of the charge multiplication process, that basically adds a peculiar noise, a detailed modeling and extensive simulations are necessary in order to properly compare this type of device to standard CCDs. This has been performed in the framework of the WLBT system,[12] and the LLLCCD implementation has been done within module PYR, but it will be distributed together with a generalization of it also to modules SHW and IMG, in a next future.

Wave-front reconstruction/command control

The standard control law implemented for now within module REC is based on a Truncated Singular Value Decomposition. The truncation threshold and the integrator gain are actually chosen by the user, but the use of the knowledge of the SNR on each of the system modes (i.e. the modes that diagonalize $D^T D$ where D is the interaction matrix) would allow to optimize the choice of gains on each of the modes without the need of any truncation of the singular values, the low SNR modes being already filtered by lower gains. This first approach, based on the OMGI (Optimal Modal Gain Integrator) concept, is being implemented. A further step will concern an optimal control law based on a Kalman filter, that will in addition permit to predict the evolution of the turbulent phase, by the use of the spatial and temporal statistics of the atmospheric turbulence.[24]

Interferometry: densified pupil modeling

This last type of new development is being implemented in the beam combiner module IBC, but already permitted some studies for the VIDA instrument.[21] An illustration of it, showing the VLTI equipped with MACAO in both Fizeau and Densified Pupil modes, is shown in Fig. 2 in terms of resulting "PSFs" in band K.

4. SOFTWARE PACKAGES DEDICATED TO OTHER THEMATICS

4.1. Image Reconstruction: The Software Package AIRY

The Software Package AIRY[13] is designed for simulating optical and near-infrared interferometric observations and/or performing subsequent image restoration/deconvolution. It has been conceived for application to LBT, but can also be used for any type of astronomical images. The modules implemented in the present version (2.1) are described in Table 2. As concerns deconvolution, AIRY includes both an implementation of the extension of the Richardson-Lucy method to multiple images and its accelerated version known as the OS-EM method. Recent developments also include super-resolution capabilities and an ulterior acceleration implementation of the Richardson-Lucy and OS-EM methods.

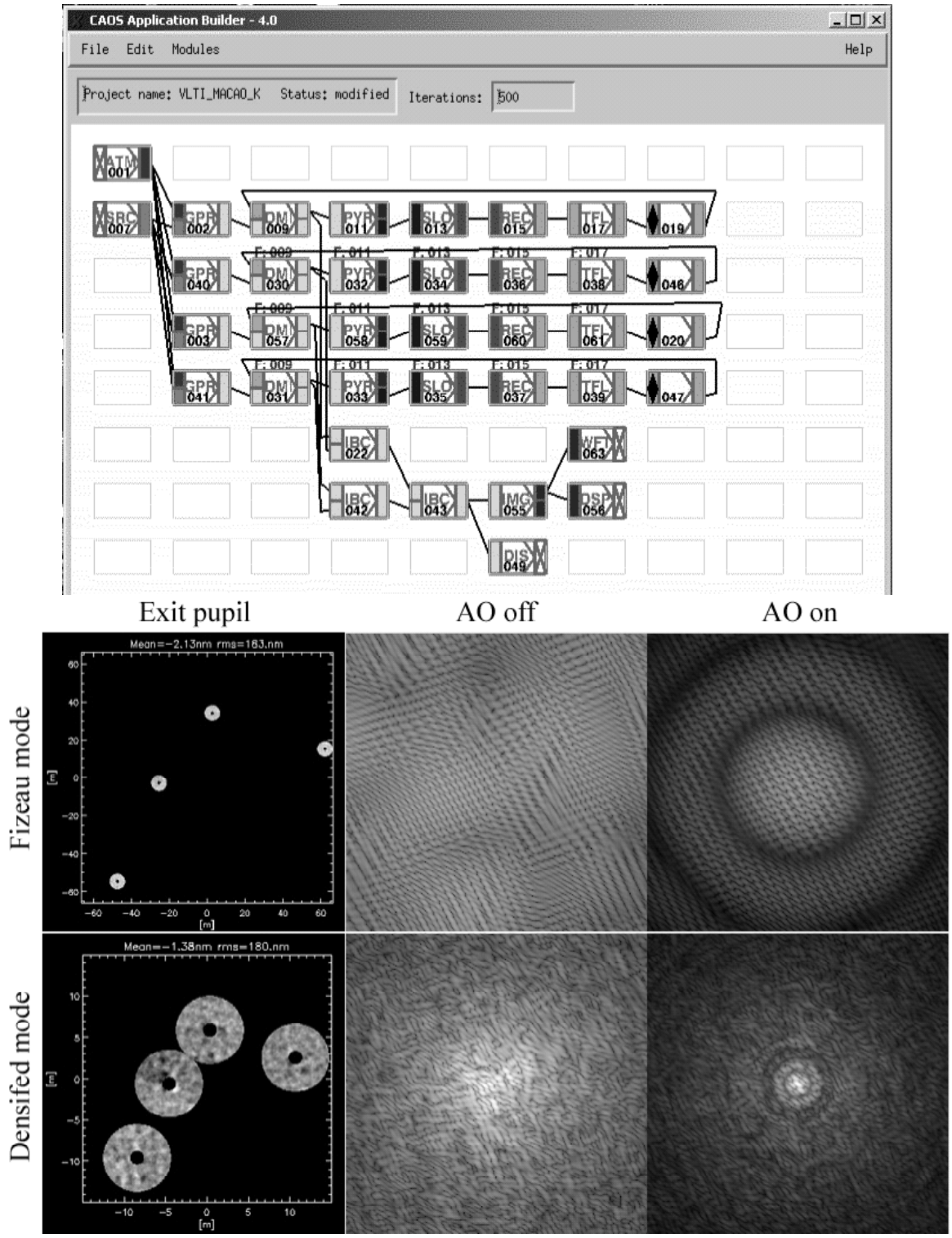

Figure 2. VIDA simulations using CAOS. Top: the project used to simulate the four UTs of the VLTI equipped with MACAO, and interferometrically recombined in Fizeau mode or Densified Pupil mode. Bottom: the pupil configuration and resulting "PSFs" corresponding to each recombination mode, with and without the AO correction provided by MACAO.

Module	Purpose
Data simulation modules	
OBJ - OBJect definition	-to define the object characteristics
CNV - object-PSF CoNVolution	-to perform convolution
ADN - ADd Noise to image	-to add background, read-out and Poisson noise
Data processing modules	
PRE - PRE-processing	-to perform image pre-processing
DEC - DEConvolution process	-to perform deconvolution (OS-EM method)
Data analysis modules	
ANB - ANalysis of Binary	-to analyse reconstructed images of binary objects
FSM - Find Star Module	-to detect stars in the reconstructed images
Other modules and utilities	
RFT - Read FiTs file format	-to read FITS images
WFT - Write FiTs file format	-to write FITS images
RSC - Restore im. Struct. Cubes	-to restore image structure cubes (XDR or FITS format)
SIM - Save IMage struct.	-to save image structure cubes (XDR or FITS format)
DSP - data DiSPlay	-to display images

Table 2. Descriptive list of the modules of the Software Package AIRY (version 2.1).

Example of application: coupling CAOS & AIRY for post-AO image reconstruction studies

Figure 3 shows a typical coupling between the Software Package CAOS and the Software Package AIRY, which is being used within our post-AO interferometric image reconstruction studies. A number of works have already been done following this basic scheme, including the limitations due to both the AO correction and the angular coverage,[5,8] the determination of computationally efficient Richardson-Lucy-like algorithms for a quick-look of the data that will be taken with the instrument LINC-NIRVANA on board LBT,[1] and the proposition of a super-resolution algorithm that basically permit to increase angular resolution capabilities up to a factor of five.[2] The complete strategies that this kind of simulation coupling permitted us to establish is presented in Anconelli et al.[3]

4.2. Multiconjugate Adaptive Optics: The Software Package MAOS, version 1.0

Most of the modules of the Software Package CAOS used in classical single-conjugate AO are being generalized to MCAO, giving birth to a new set of modules specialized in MCAO, and hence constituting a new Software Package: MAOS (Multiple-reference multiconjugate Adaptive Optics Simulations). Table 3 shows a list of the various modules that are being built for this package.

Table 3. Descriptive list of the modules of the Software Package MAOS (version 1.0).

Module	Purpose
MSC - Multiple SourCe definition	-to characterize the guide stars (GS's) asterism
MGP - Multiple Geometrical Propagator	-to geometrically propagate the light from a field of objects
MMG - Multiple iMaGing device	-to make an image of a field of objects
COM - COMbine measures	-to combine the sensors measures (classical MCAO scheme)
COA - CO-Add pyramid images	-to optically co-add pyramid images (LO MCAO sheme)
CON - CO-add Numerically	-to numerically co-add sensor images (LO MCAO sheme)
DMC - Deformable Mirror Conjugated	-to apply the correction at a given conjugated altitude

Preliminary versions of some of these modules were already used,[31] and we hope to soon be able to deliver

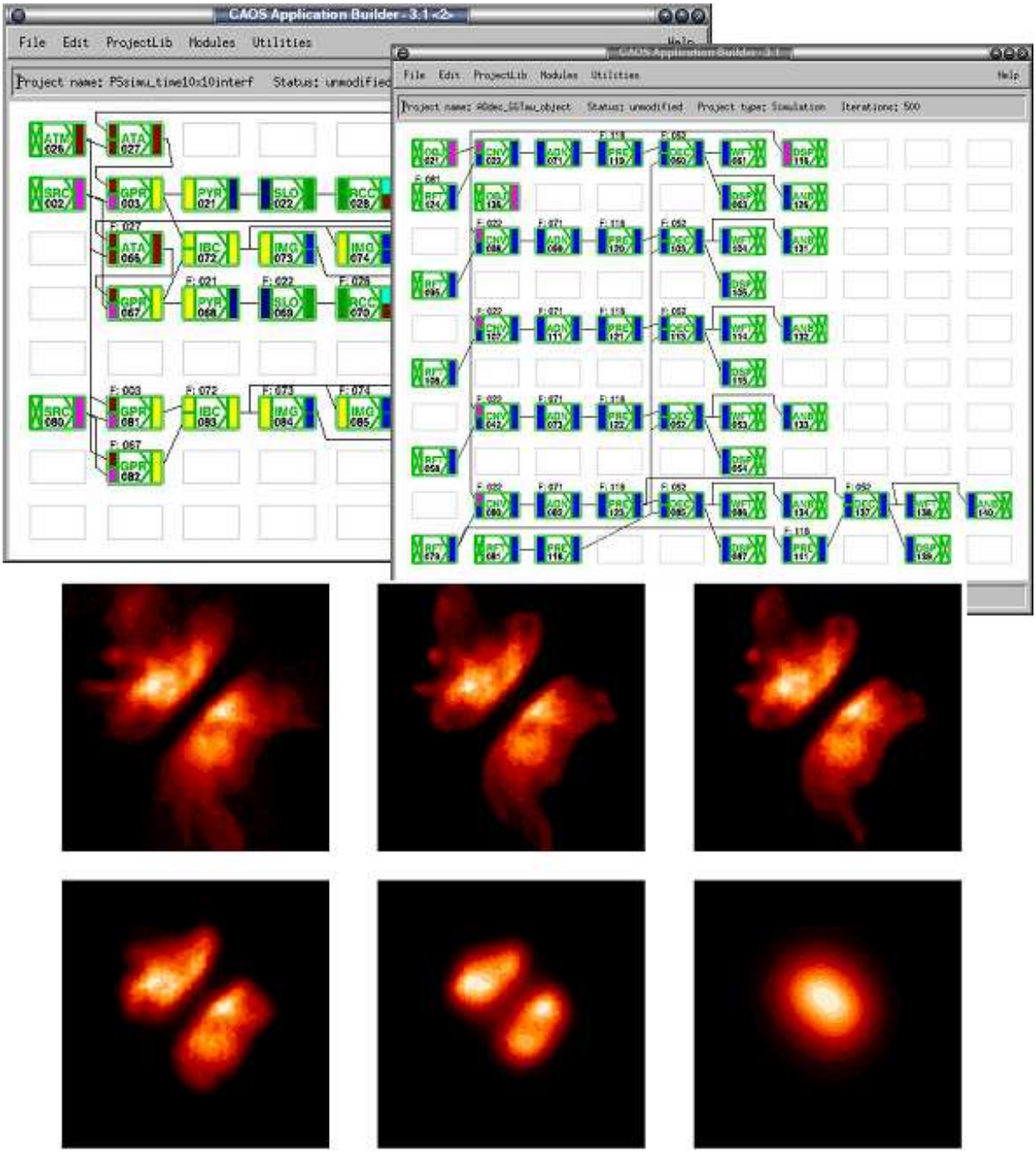

Figure 3. Top: simulation projects within the CAOS Application Builder and using in one hand the modules of the Software Package CAOS (background) for the AO-corrected LBT interferometric PSFs, and in the other hand the modules of the Software Package AIRY (foreground) for simulation and deconvolution of the resulting images. Bottom: post-AO image reconstruction of a *Butterfly Star*-like object for the different AO-correction qualities simulated. From left to right and from top to bottom: the object, and the image reconstructed for five cases of decreasing AO correction. Extension of the images is $\sim 0\farcs85$, imaging band is K ($2.2\,\mu$m).

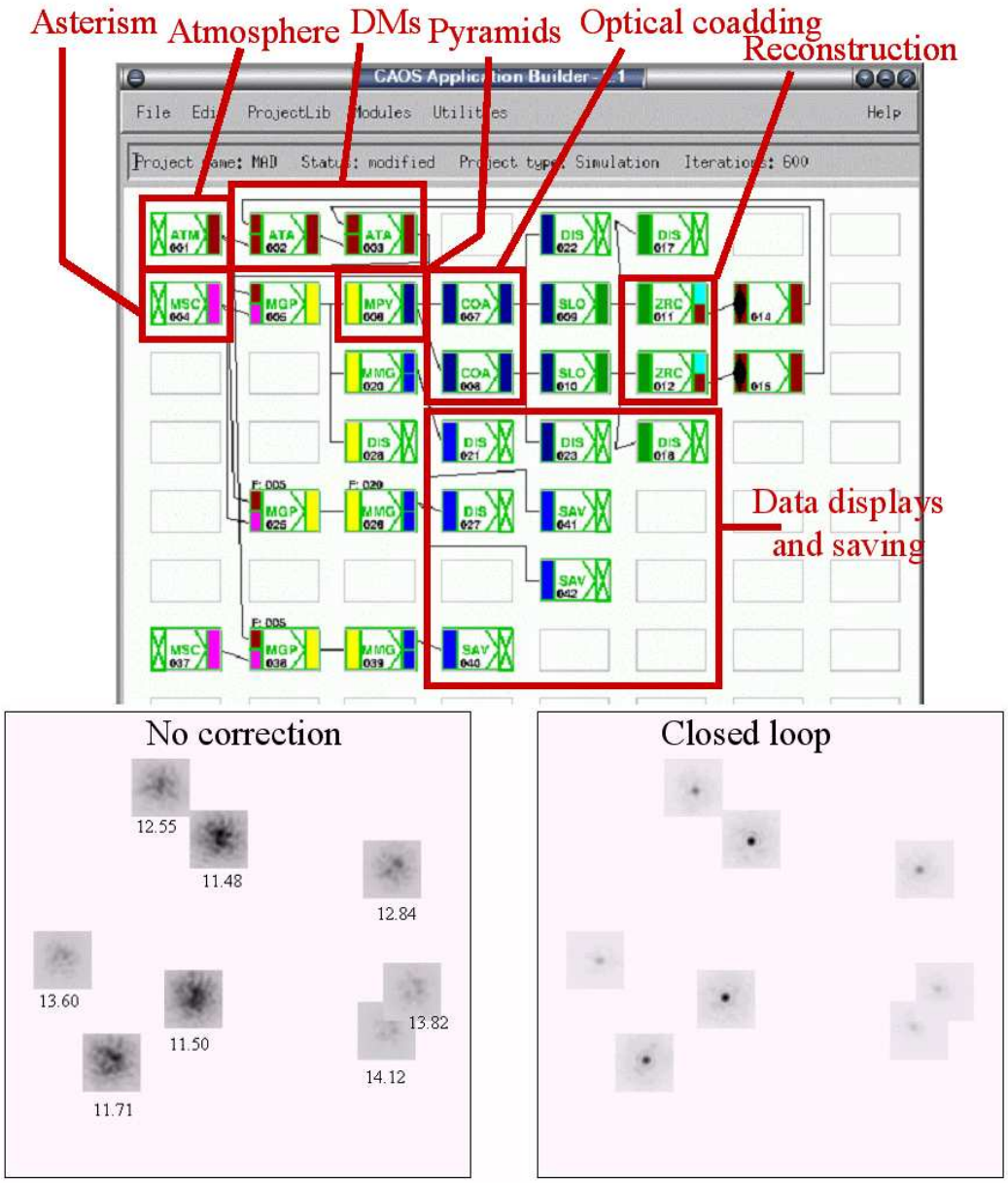

Figure 4. Top: the CAOS Application Builder showing a typical simulation design using the modules of the Software Package MAOS for simulating the ESO MCAO demonstrator MAD, in LO mode. Bottom: the result of the simulation in terms of PSF shape in the sensing field (sensing magnitudes are also marked).

version 1.0 of the package to the whole community (a β version is already distributed together with the Software Package CAOS). Fig. 4 illustrates the use of the package in the case of the MCAO demonstrator MAD used in layer-oriented (LO) mode.

5. PARALLELIZATION OF CAOS

Simulation programs are typically very CPU intensive and are usually well suited to exploit High Performance Computing (HPC) techniques. On the other hand, the usual life cycle of typical simulation programs is characterized by a long development phase followed by a small numbers of runs; i.e.: the careful coding and optimization effort which is needed to properly tune the code for running on a parallel architecture, is usually not very rewarding. For this reason we have explored two approaches which avoid the need to write specific parallelization code and yet allow us to take advantage of a beowulf-like parallel machine.

The first one is based on the concept of the "number crunching engine": a few numerical functions have been selected for an optimized parallel implementation. Whenever one of such functions is called in the simulation code a software stub provides for getting the service done by the parallel version of the function. Within this framework, we have implemented `par_idl`, a parallel virtual machine that coordinates the parallel execution and manages the message passing.

A software layer is also needed to let IDL processes communicate with `par_idl`. At the moment, no other implementation of IDL is available for distributed memory parallel machines. In `par_idl`, computational concurrent paradigm that is used is *SPMD* (Single Program Multiple Data), in which multiple instances of the same program run on different computers and execute the same operation on different data, loosely synchronizing with message passing. In order to obtain the parallel version and to have an efficient implementation, we have chosen in particular two mechanisms available in IDL for interprocess communications and management. The first enables the execution of IDL macros by another program, i.e. the ability for a program written in `C` to call a macro written in IDL. Different functions are provided to activate the IDL parser, load the function, and execute its instructions and free resources. In the same way, it is possible to call external functions, for example implemented in C, and execute them within an IDL macro. Each process that is executing an IDL macro is connected to a proxy, that provides the tool for the IDL process to exchange data with the parallel virtual machine. Each time a macro has to send data to an instance of the same macro running on a different node, it calls an external function of the proxy which provides the meaning to share such data with the underlying software `par_idl` managing all communications. At the same time the IDL process waiting to receive the message, waits until the proxy receives the data, exchanged by the parallel virtual machine. The communication overhead is affected by both the inter-processor communication mechanism used on each single computer and the message passing library. Nevertheless, the use of existing message passing libraries, such as the ones implementing MPI standard (`http://www.mpi-forum.org/`), makes it easier to develop robust and portable software. That reasoning prevented us from using the internal RPC mechanism provided by IDL, since, in that case, we had to build a robust message exchange mechanism and re-implement all collective and one-to-one communication primitives, that can be found in any MPI library.

The work reported goes beyond a simple interface to MPI, since it is possible to integrate in CAOS other tools for application development. The availability of the virtual machine, coordinating work on the parallel computer, permits the integration of parallel libraries: each call to a module in a library corresponds to a data sharing between the IDL and `par_idl` software layers and to a subsequent call to the library object. That provides a way to integrate existing libraries even without IDL source code; the drawback is that the library functions have to be called by the parallel virtual machine, that means that each API has to be incorporated within `par_idl`.

The second approach, which is still in a conceptual phase, takes advantage on the fact that the CAOS Application Builder already provides automated code generation. By adding a few tools to evaluate load balancing we may add the capability to automatically generate parallelization code targeted to a specific multiprocessor architecture. The rationale behind this last approach has been discussed elsewhere,[17] and the actual implementation will start in the near future.

6. FINAL REMARKS

The CAOS Application Builder and the Software Package CAOS can be downloaded from the dedicated web page `http://www.arcetri.astro.it/caos/`. Please subscribe to the dedicated mailing-list when having downloaded

the package. The Software Package AIRY can be obtained by visiting `http://dirac.disi.unige.it/`. The Software Package MAOS, version 1.0, is not yet available, but a β version of it can be obtained when downloading the Software Package CAOS.

The mailing-list dedicated to the Software Package CAOS currently features 52 registered users, while the one dedicated to the Software Package AIRY currently features 12 registered users.

REFERENCES

1. B. Anconelli, M. Bertero, P. Boccacci, M. Carbillet, H. Lanteri, *submitted to* Astron. Astrophys., 2004a
2. B. Anconelli, M. Bertero, P. Boccacci, M. Carbillet, *submitted to* Astron. Astrophys., 2004b
3. B. Anconelli, M. Bertero, P. Boccacci, M. Carbillet, H. Lanteri, S. Correia, SPIE Proc. 5491, 2004c
4. C. Arcidiacono, E. Diolaiti, M. Tordi, et al., *to appear* in App. Optics, 2004
5. M. Carbillet, S. Correia, P. Boccacci, M. Bertero, Astron. Astrophys. **387** (2), 2002a
6. M. Carbillet, B. Femenía, S. Esposito, et al., ESO Conf. & Workshop Proc. **58**, É. Vernet, R. Ragazzoni, S. Esposito, and N. Hubin Eds., 2002b
7. M. Carbillet, C. Vérinaud, S. Esposito S., et al., SPIE Proc. **4839**, P. L. Wizinowich & D. Bonaccini Eds., 2003a
8. M. Carbillet, S. Correia, M. Bertero, P. Boccacci, SPIE Proc. **4840**, W. A. Traub Ed., 2003b
9. M. Carbillet, A. Riccardi, S. Esposito, these proceedings, 2004a
10. M. Carbillet, C. Vérinaud, B. Femenía, A. Riccardi, L. Fini, submitted to MNRAS, 2004b
11. M. Carbillet, EAS Publ. Series, C. Aime & R. Soummer Eds., in press, 2004c
12. M. Carbillet, in preparation for App. Optics, 2004d
13. S. Correia, M. Carbillet, P. Boccacci, et al., Astron. Astrophys. **387** (2), 2002
14. S. Esposito, A. Tozzi, A. Puglisi, E. Pinna, P. Stefanini, L. Fini, P. Salinari, these proceedings, 2004
15. B. Femenía & N. Devaney, Astron. Astrophys. **404**, 2003
16. L. Fini, M. Carbillet, A. Riccardi, ASP Conf. Series **238**, F. A. Primini & F. R. Harnden Eds., 2001
17. L. Fini & M. Carbillet, ASP Conf. Ser. **295**, H. E. Payne, R. I. Jedrzejewski, and R. N. Hook Eds., 2003
18. I. Foppiani, C. Baffa, V. Billiotti, G. Bregoli, G. Cosentino, E. Giani, S. Esposito, B. Marano, P. Salinari, Proc. SPIE 4837, J. M. Oschmann & L. M. Stepp Eds., 2002
19. L. Jolissaint & J.-P. Véran, ESO Conf. & Workshop Proc. **58**, É. Vernet, R. Ragazzoni, S. Esposito, and N. Hubin Eds., 2002
20. R. Koehler, S. Hippler, M. Feldt, R. Gratton, D. Gisler, R. Stuik, J. Lima, these proceedings, 2004
21. O. Lardière, D. Mourard, F. Patru, A. Labeyrie, J. Dejongue, F. Martinache, M. Carbillet, SPIE Proc. 5491, 2004a
22. O. Lardière, M. Carbillet, A. Riccardi, P. Salinari, these proceedings, 2004b
23. M. Le Louarn, J. Braud, E. Fedrigo, et al., these proceedings, 2004
24. B. Le Roux, J.-M. Conan, C. Kulcsár, et al., J. Opt. Soc. Am. A **21** (7) 2004
25. D. Looze, S. Hippler, M. Feldt, these proceedings, 2004
26. A. Riccardi, G. Brusa, M. Xompero, D. Zanotti, these proceedings, 2004
27. G. Sacco, R. Pallavicini, P. Spanò, M. Andersen, K. G. Strassmeier, these proceedings, 2004
28. M. Smith, E. Steinbring, S. Roberts, G. Heriott, J. Dunn, J.-P. Véran, D. Kerley, these proceedings, 2004
29. E. Steinbring, S. M. Faber, B. A. Macintosh, D. Gavel, E. L. Gates, in preparation for PASP, 2004
30. C. Vérinaud, Opt. Letters **27** (7), 2002
31. C. Vérinaud, Arcidiacono C., Carbillet M., et al., SPIE Proc. **4839**, P. L. Wizinowich & D. Bonaccini Eds., 2003a
32. C. Vérinaud & M. Carbillet, EAS Publ. Series **8**, 209, C. Aime & R. Soummer Eds., 2003b
33. C. Vérinaud, M. Le Louarn, V. Korkiaskoski, J. Braud, M. Carbillet, these proceedings, 2004