Third AO4ELT Conference - Adaptive Optics for Extremely Large Telescopes
Florence, Italy. May 2013
ISBN: 978-88-908876-0-4
DOI: 10.12839/AO4ELT3.13173

SIMULATING ASTRONOMICAL ADAPTIVE OPTICS SYSTEMS USING YAO

François Rigaut[1a] and Marcos van Dam[2,3]

[1] The Australian National University, RSAA, Mount Stromlo Observatory, Cotter Road, Weston Creek ACT 2611, Australia
[2] Gemini Observatory, c/o AURA, Casilla 603, La Serena, Chile
[3] Flat Wavefronts, PO BOX 1060, Christchurch 8140, New Zealand

Abstract. In this paper, we describe YAO, an adaptive optics simulation software package that has been around and used by the community for the past twelve years. It offers a very comprehensive set of functionalities, covering many different wavefront sensors, deformable mirrors, reconstructors and control laws. CPU intensive core functionalities are coded in C, thus optimized for speed. The glue is provided by yorick, which allows quick development, easy debugging and plotting functionalities. In this paper, we describe YAO, including installation, performance and documentation, and give examples of scripting capabilities.

1 Introduction

Adaptive Optics (AO) systems are at the heart of the coming Extremely Large Telescopes generation. Given the importance, complexity and required advances of these AO systems, being able to simulate them faithfully is key to their success, and thus to the success of the ELTs. The type of AO systems envisioned to be built for the ELTs cover most of the AO breeds, from Natural Guide Star (NGS) AO to multiple guide star Ground Layer (GLAO), Laser Tomography (LTAO) and Multi-Conjugate AO (MCAO) systems, with typically a few thousand actuators. This represents a large step up from the current generation of AO systems, and accordingly a challenge for existing AO simulation packages. This is especially true as, in the past years, computer power has not been following Moore's law in its most common understanding; CPU clocks are hovering at about 3 GHz. Although the use of super computers is a possible solution to run these simulations, being able to use smaller machines has obvious advantages: cost, access and environmental issues. By using optimised code in an already proven AO simulation platform, we are able to run complex ELT AO simulations on very modest machines, including laptops.

In this paper, we describe YAO, a platform for running a variety of astronomical AO simulations. We discuss its architecture, its capabilities, the ELT-specific challenges and optimisations, and finally its performance.

As an example, execution speed ranges from 5 iterations per second for a 6 LGS 60×60 subaperture Shack-Hartmann Wavefront sensor Laser Tomography AO system for the 25m Giant Magellan Telescope (GMT) (including full physical image formation and detector characteristics) up to over 30 iterations/s for a single NGS AO system for the same GMT. [I would be tempted to give one concrete examples somewhere later in the paper. e.g., number of guide stars, pixel size in m, number of science images and wavelengths, etc,

[a] francois.rigaut@anu.edu.au

and show simulation time. what we have here is not useful to be able to compare with speeds of other simulation packages, for example]

Yao has been around since 2001 and has been used extensively by the AO community to design or investigate the performance of many systems (the following list is non-exhaustive): GeMS [2], Vasao [16], Kapao [17], Gmt Ltao [5], Gmt Glao [6,14], CANARY [3], Altair [7], Gravity [9], Imaka [11], Guieloa [15], Siding Spring AO [10], other new techniques [8] and research [12,13] and even in some cases integrated into the system operation (e.g. GeMS [4]). As such, it has been under the scrutiny of many teams and has become a proven and well debugged platform.

2 The YAO package

2.1 History

Yao is an open source AO software simulation package. It has evolved from an IDL simulation tool written in 1992-1999 by F. Rigaut while at CFHT. Then aptly and originally named "`aosimul`", it already boasted several type of wavefront sensors and deformable mirrors. There was extensive cross-calibration and comparison work with other AO simulation packages [1], providing more confidence in the simulation results. Following the lack of plans of RSI to port IDL to OsX in 1999, the package was ported to yorick, renamed Yao (for Yorick Adaptive Optics), and then expanded for use in many projects, by a number of contributors (Francois Rigaut, Marcos van Dam, Damien Gratadour, Ralf Flicker, Aurea Garcia-Rissman and Eric Gendron). It is now mostly maintained and developed by the authors (lots of features added in the past 2 years within the GMT simulation efforts). It is open source (GPL license) and lives on github (http://github.com/frigaut/yao).

2.2 Installation

The latest version is 5.3 (as of October 2013). Yao is a popular yorick plugin and as such can be installed from many repositories: Debian/Ubuntu/Mint/Archlinux for linux, macports for OsX. Dependencies include few other yorick plugins (yutils, imutil, soy) and libraries (FFTW, python & gtk for the GUI). Note that sometimes distribution repositories lag in releasing the latest package versions. Alternatively, one can always install from source ([18]).

3 YAO feature set

Yao has a very extensive feature set. Features have been aggregated as needed, often by external contributors. Below is a subset of features:

- Wavefront sensors features:
  - Shack-Hartmann, Curvature, Pyramid, Tip-Tilt, modal (Zernike/KL/Disk Harmonic),
  - Photon and read out noise, bias and flat field errors,
  - Extended objects for some WFS types,
  - Natural or Laser Guide Stars (or mix of both),
  - Arbitrary number of WFSs (ON or OFF axis), can mix types (with certain limitations),

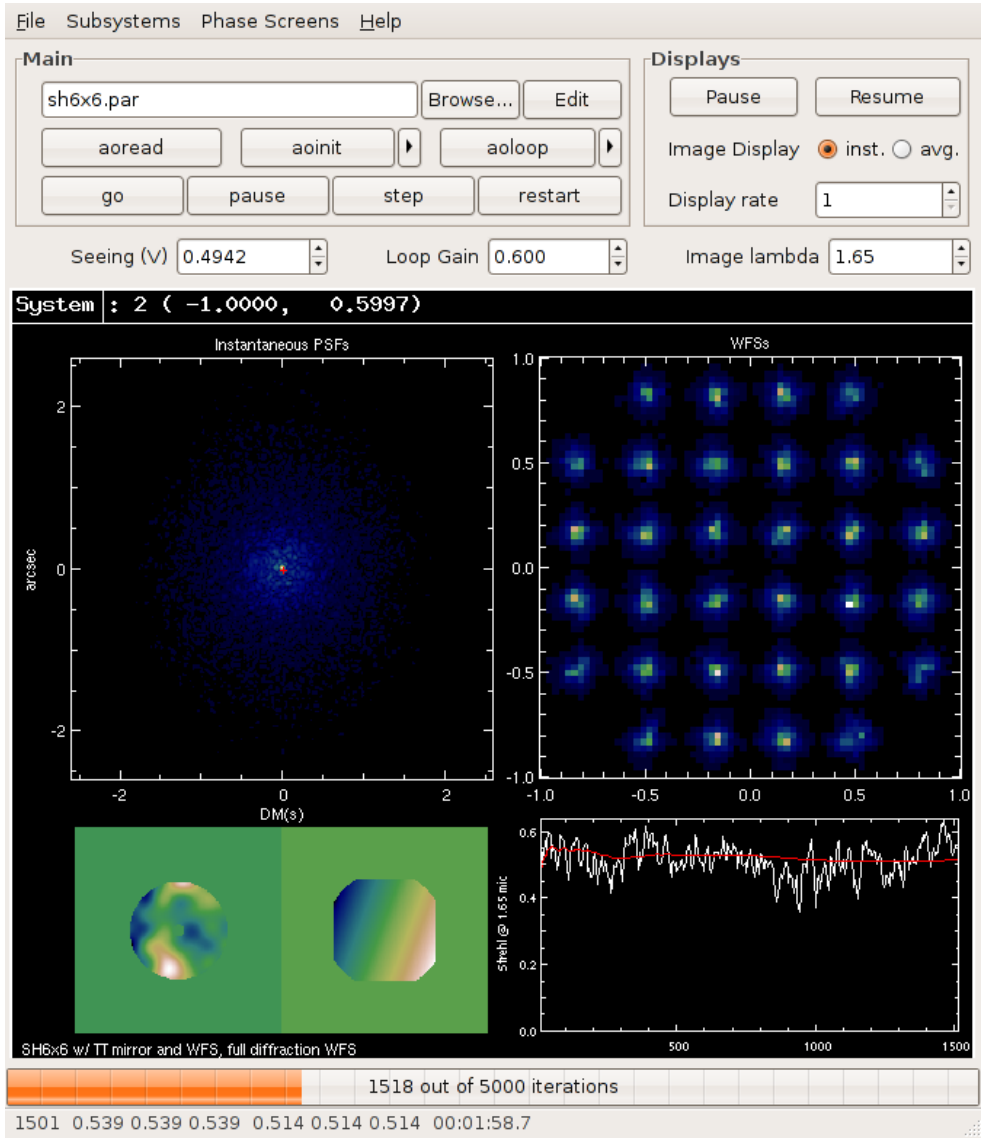

Fig. 1. The YAO graphic window, with the basic GUI. The four graphical areas show the PSF(s) in the upper left, some kind of WFS display (spots for SHWFS, extra-focal image for curvature, four pupils for a Pyramid WFS) in the upper right, DM surface in the lower left and Strehl ratio versus time in the lower right. The YAO GUI, although it only offers access to a limited number of functionalities, is a convenient way to interact with yao to play with the system. Changes through the GUI are taken into account live. The GUI is gtk based.

  - Shack-Hartmann:
    - Full physical (diffraction) model, many parameters adjustable (see right panel of Fig 2 and Fig 3)
    - or simple gradient average for quick and dirty estimates.

- Deformable Mirrors:
  - Stackarray, Curvature, Segmented, Tip-Tilt, modal (Zernike/KL/DH),
  - Hysteresis, saturation, misregistration (see Fig 3), extrapolated actuators,
  - Multiple DMs supported, with selectable conjugation altitude.
- An arbitrary number of WFSs and DMs can be used, with the possibility of mixing types. It is therefore possible to simulate single DM systems, as well as single non-zero

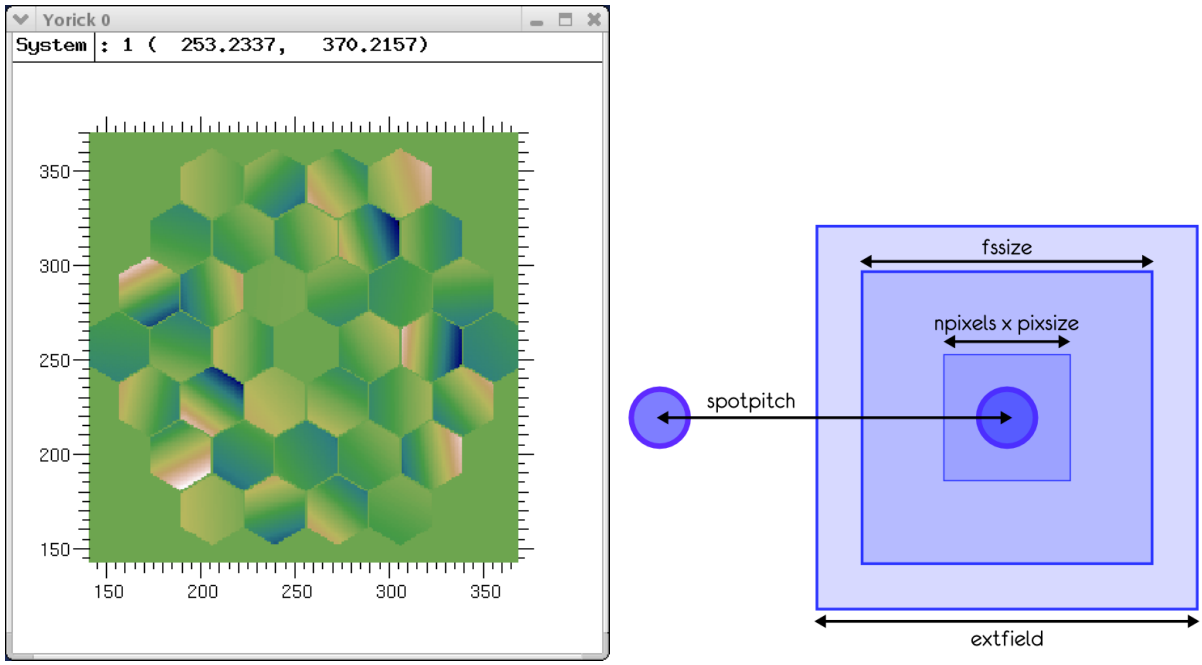

Fig. 2. Left: YAO offers segmented pupils, contiguous (TMT/EELT) or not (LBT/GMT). Right: Even though most of the parameters are optional and have sane default values, Shack-Hartmann WFS configuration include lots of choices.

conjugate, GLAO and MCAO systems. WFS and DM can be partitioned in subsystems to isolate behavior, as can typically be found in LGS high order and NGS TT control in LGS systems.

- It supports Natural and Laser Guide Stars (or a mix). LGS support include spot blurring and elongation with a user supplied LGS profile (there are templates), uplink jitter correction and more.
- Laser Guide Stars:
  - Elongation, LLT position, uplink seeing and jitter,
  - Uplink TT stabilisation,
  - Rayleigh fratricide for multi-beam systems (SHWFS only).
- Reconstructors and control
  - Least square, MMSE/MVR and MMSE sparse (with regularisation),
  - Control law with up to 10 numerator and denominator coefficients,
  - Closed-loop, open-loop or pseudo open-loop Control,
  - Adjustable loop delay (integer multiple of exposure time).
- Adjustable multi-point, multi-wavelength performance estimate
- Easily add your own pupil, WFS or DM through a user function with well defined APIs
- User hooks within the main loop or for reconstructor generation. That is, call to functions (like `user_end_go()`) are coded and the function is executed if it has been defined by the user.
- Multi-layered atmosphere, geometrical propagation only,
- User supplied vibration spectrum can be used to simulate vibrations,
- Static aberrations on the WFS, science or common path,

- Lots of internal variables are accessible from the outside for debugging/adding new features,
- Extensive documentation (see below yao::manual),
- Good support (for free, so not always lightning fast...),
- Easy scriptability to investigate parameter space.

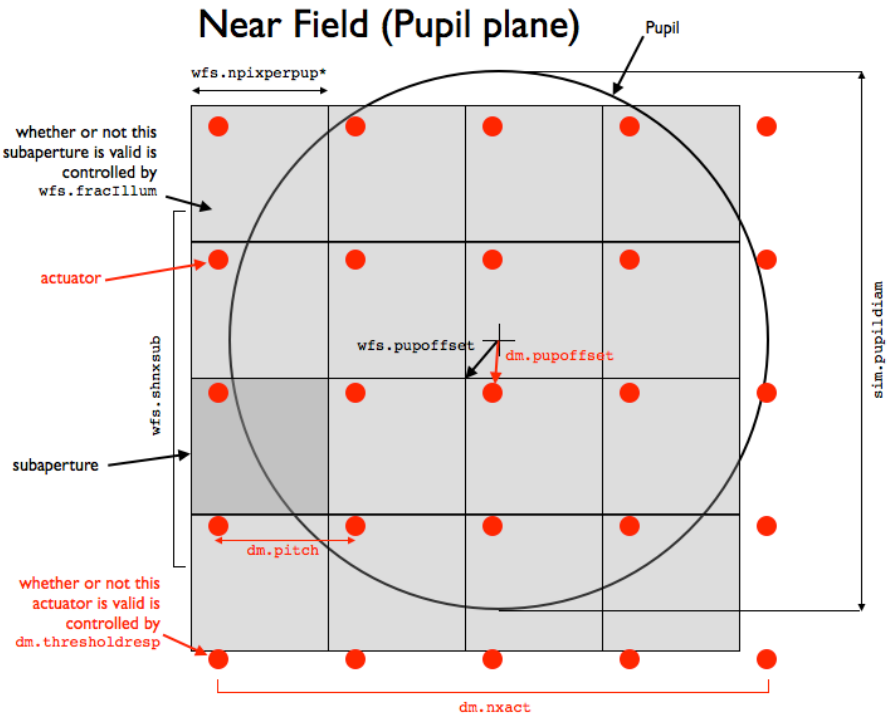

Fig. 3. High level summary of a typical Stack-array DM / Shack-Hartmann WFS configuration, with fine tuning on enabled subapertures and actuators, mis-registration, etc.

4 Performance

4.1 YAO is fast!

YAO is fast for a general-purpose simulation package. The speed of course can not approach the speed of dedicated real-time platforms.

Below are the results of running "`yorick -i test-all.i`" using `yao 5.2.0` on Wed May 22 16:09:05 2013. The machine is a laptop (retina macbook pro) with the following characteristics: 2.6GHz intel core i7, 16GB 1600Hz DDR3 RAM, nvidia GT650M, running macos 10.8.3, yorick 2.2.02. Below, "iteration/s" includes the turbulent phase computation, wavefront sensing, reconstruction, calculation of the DM(s) shape and PSF(s)/performance estimation.

```
Parfile Name                                              iter/s Display?  Strehl
test    Simple SH6x6 w/ TT mirror, full diffraction WFS  315    ON        0.45@1.65mic
test    Simple SH6x6 w/ TT mirror, full diffraction WFS  387    OFF       0.45@1.65mic
test1   Altair LGS like, full diff WFS, spot elongation   92    OFF       0.59@1.65mic
test2   SH 12x12, quick method, separate 4cell and TTM   485    OFF       0.79@1.65mic
```

test3	GLAO example, 3 8x8 SHWFS, w/ disk harmonics	73	OFF	0.05@2.23mic
test4	Simple Curvature 36 actuators with noise	870	OFF	0.58@2.20mic
test5	2.5 metre telescope w/ Pyramid WFS	108	OFF	0.84@2.20mic
test6	Disk Harmonic WFS \& DM (45 modes)	1023	OFF	0.50@1.65mic
test7	LGS SH12x12, Extended WFS (11 npt profile)	38	OFF	0.62@1.65mic
test8	Simple SH6x6 w/ TT mirror, geometrical WFS	1838	OFF	0.36@1.65mic

4.2 YAO speed

Yorick is an interpreted language, so how can YAO be so fast? Because the critical routines have been coded in C. Yorick is used as a convenient "glue" between lower levels optimised C calls, in parts of the code where what counts is not speed, but ease of development, graphical display capabilities, code compactness. In our opinion, this uses the best of both worlds. A 64×64 Shack-Hartmann (full diffraction model) system runs at about 10 iterations/s for an 30-m telescope. A 188 curvature system runs at over 120 iterations/s. Medium-size AO systems for ELTs can also be simulated. Simulating the GMT LTAO system (6 LGS WFS each 60×60 subapertures, including spot elongation) runs at 6 iterations/seconds and take slightly over 1.5 GB or RAM.

Yao can be used in parallel mode, using the yorick svipc plugin. On modern hardware, it is now possible to simulate up to 200×200 SHWFS (1 iteration/s). A 64×64SHWFS runs at 23 iterations/s).

5 Using YAO

5.1 A basic YAO session

In Yao, a system is defined by a parameter file, which is just a yorick script file, in which the user assigns values to various structures needed to define the AO system. There are 10 structures: `atm` (atmosphere, to define the atmospheric turbulence, r_0, Cn^2 etc), `wfs` (wavefront sensor), `dm` (deformable mirror), `tel` (telescope), `loop` (loop and control), `mat` (reconstructor), `gs` (guide star), `target` (PSF estimation) are the main ones. These structures can be made into structure arrays if the system has several WFSs or DMs. Structures are defined in the file `yao_structures.i`, with short explanations for all structure members. Users are encouraged to browse it. Of course, starting from scratch to define a yao parfile is somewhat daunting, but Yao comes with many examples of parameter files that the user can use as a starting point. There are parameter files examples for SHWFS, curvature systems, with NGS or LGS, etc.

A basic Yao session consist in going through four function calls. Typically, the first call is to `aoread()` which reads the parfile and populates the structures. Next, `aoinit()` initializes the system: WFS, DMs, etc. Then, `aoloop()` and `go()` starts the actual close loop Monte-Carlo simulation. The next subsection (5.2) illustrates this.

5.2 Scriptability

One of the most popular use of AO simulations is to optimize system parameters when designing a system. Using scripts, Yao makes it easy to probe parameter space. Below is an example of a script to find the best gain for several guide star magnitude value. This

can of course easily be extended to include more dimensions, that is other parameters (although one has to be careful that the number of points to probe grows very fast with the number of dimensions). This is essentially the `yao_loop_example.i` yorick script included in the yao distribution, stripped of unessential parts (e.g. graphical display).

```
require,"yao.i";

// read out parfile
aoread,"test.par";

// you can modify parameters in here
atm.dr0at05mic = 35; // be more gentle

// define vector on which we want to loop and final strehl array
// we want to estimate performance for 3 values of the guide star
// magnitude and 4 values of the loop gain (for instance)
gsmagv = [6,9,12];
gainv  = [0.01,0.1,0.5,1.0];
strehlarray = array(0.,[2,numberof(gsmagv),numberof(gainv)]);

// loop on gsmag and gain
for (ii=1;ii<=numberof(gsmagv);ii++) {
  for (jj=1;jj<=numberof(gainv);jj++) {
    wfs(1).gsmag=gsmagv(ii);
    loop.gain=gainv(jj);
    // it's safer, but not always necessary, to call again
    // aoinit (here for gsmag). some parameters do not need it.
    aoinit,disp=1;
    aoloop,disp=1;
    go, all=1;
    // after_loop is now called automatically at last iter of go()
    strehlarray(ii,jj) = strehllp(0); // fill in result array
    // strehllp is the vector of of long exposure Strehl.
    // Other diagnostics filled by after_loop() are:
    // strehl: Vector of Strehl for each images position and
    //         wavelength (long exposure)
    // fwhm and e50: FWHM and 50% encircled energy for each
    //         position and wavelength.
  }
}
```

6 Documentation

This article has just reported on a very small subset of YAO's capabilities. There is a fair amount of documentation on-line. The top YAO website is at http://frigaut.github.io/yao/ and contains guides to install and configure the package, as well as some news (blog). Installation is supported on linux and OsX. YAO should also work on Windows, but this is not supported and has never been tested.

There is a fairly extensive YAO manual at http://frigaut.github.io/yao/manual.html, which addresses how to configure parfiles, how to control features (global system, wfs, dms, etc), provide help on scripting and how to run a yao session. The documentation is extensive; it could be updated more often however, and some of the recent features

are not always documented as well as they should. Contributions to YAO from the AO community are welcome.

7 Conclusion

YAO is a proven, reliable, full featured adaptive optics simulation platform. It is fast and flexible. It has already been used as support for many adaptive optics instrument design, and has been cross calibrated with other packages. YAO is open source, and has had many contributors along its 12 years lifetime. It can easily be hacked to include a user favourite feature. YAO is being used to simulate a large number of AO systems, from small to extremely large.

References

1. Rigaut, F., Ellerbroek, B.L. and Northcott, M.J., Applied optics **36** 13, (1997)
2. Neichel, B., Rigaut, F., Bec, M. and Garcia-Rissmann, A., in 1st AO4ELT Conference, (2010)
3. Vidal, F., Gendron E. and Rousset, G., JOSA A 27.11 (2010), A253-A264
4. Rigaut, F., Neichel, B., Bec, M., Boccas, M., Garcia-Rissmann, A. and Gratadour, D., in 1st AO4ELT Conference, (2010) pp. 1-6.
5. Van Dam, M. A., Hinz, P. M., Codona, J. L., Hart, M., Garcia-Rissmann, A., Johns, M. W., ... and Rigaut, F., SPIE proceeding **7736** (2010)
6. Hinz, P. M., Bouchez, A., Johns, M., Shectman, S., Hart, M., McLeod, B. and McGregor, P., SPIE proceeding **7736** (2010)
7. Christou, J. C., Neichel, B., Rigaut, F., Sheehan, M., McDermid, R. M., Trancho, G., ... and Walls, B., SPIE proceeding **7736** (2010)
8. Perrin, G., Lacour, S., Woillez, J. and Thibaut, E., Monthly Notices of the Royal Astronomical Society **373(2)** (2006) pp747-751.
9. Clénet, Y., Gendron, E., Rousset, G. and Hippler, S., SPIE proceeding **7736** (2010)
10. Goodwin, M., Jenkins, C. and Lambert, A., Publications of the Astronomical Society of Australia, **30**, (2013) p10
11. Lai, O., Cuillandre, J. C., Chun, M. R., Carlberg, R. and Richer, H. B., Optical Turbulence: Astronomy Meets Meteorology, eds. E. Masciadri, M. Sarazin, ESO, (2009) p291-298.
12. Gratadour, D., Gendron, E., Rousset, G. and Rigaut, F., in 1st AO4ELT Conference, (2010)
13. Velur, V., Flicker, R. C., Platt, B. C., Britton, M. C., Dekany, R. G., Troy, M., ... and Hickey, J., SPIE proceeding **6272** (2006)
14. Hinz, P. M., Brusa, G., Vaitheeswaran, V., McMahon, T., Connors, T., Knox, R., ... and Montoya, M., SPIE proceeding **8447** (2012)
15. Watson, A. M., Cuevas, S., Sánchez, B., Cantó, J., Chapa, O., Flores, R., ... and Del Valle, D. RevMexAA (Serie de Conferencias) **28** (2007), pp73-81
16. Lai, O., Veillet, C., Salmon, D., Ho, K., Baril, M. R., Barrick, G. A., ... and de Chatellus, H., SPIE proceeding **7015** (2008)
17. Severson, S. A., Choi, P. I., Contreras, D. S. et al, SPIE proceeding **8617** (2013)
18. https://github.com/frigaut/yao