Object–Oriented Matlab® Adaptive Optics Toolbox

R. Conan* and C. Correia†

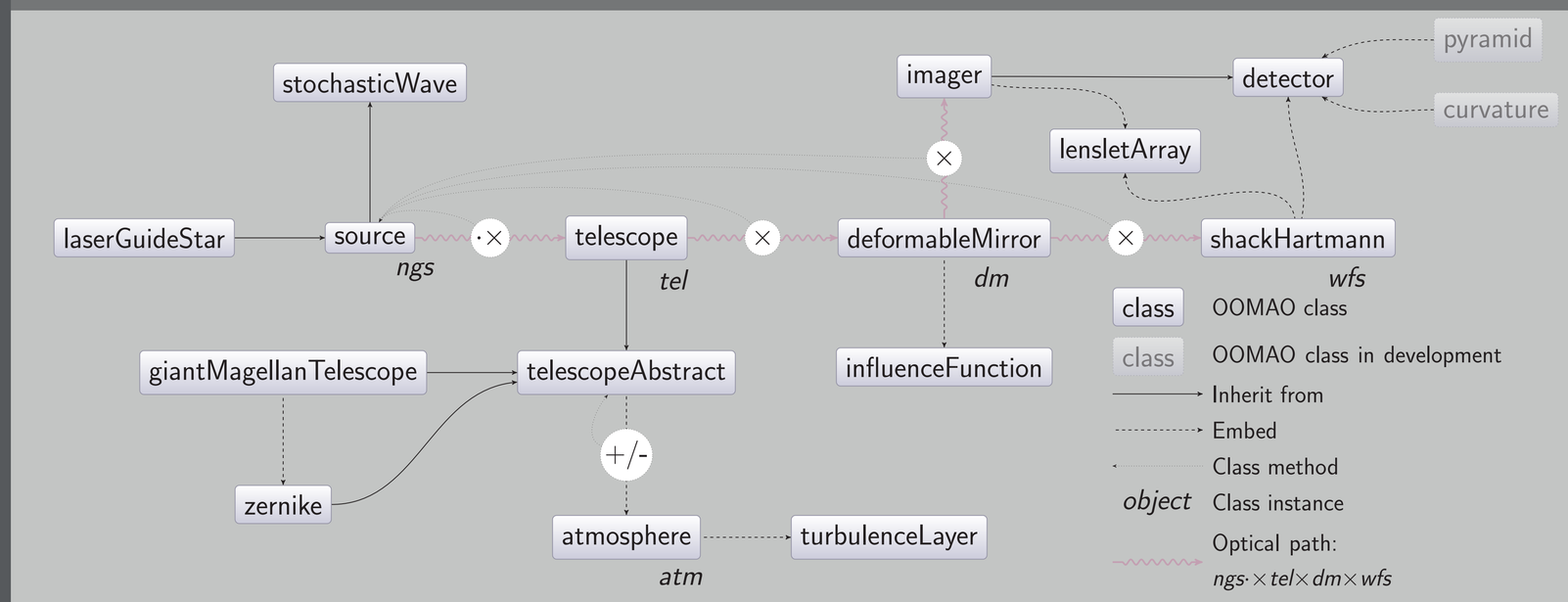
* RSAA, The Australian National University, Weston Creek, ACT 2611, Australia

† Centre for Astrophysics, University of Porto, Rua das Estrelas 4150-762 Porto, Portugal

OOMAO

OOMAO is a *Matlab*® toolbox to model Adaptive Optics systems;
OOMAO is object–oriented with classes for the AO components and for the stochastic description of the optical turbulence;
OOMAO extends Matlab vectorization capability to AO modeling;
OOMAO optimizes the use of the computer resources by relying of special matrices like sparse and Toeplitz matrices;
OOMAO uses Matlab multi–threaded native routines and its own parallel algorithms;
OOMAO can be used to model closed–, open– and pseudo–open–loop, NGS and LGS, "classical" and tomographic AO systems;
OOMAO is used in the control software of the MOAO demonstrator RAVEN at the Subaru telescope;
OOMAO has been used to define and to estimate the performance of the LTAO system of the Giant Magellan Telescope.

OOMAO Class Tree



Adaptive Optics Modeling with OOMAO

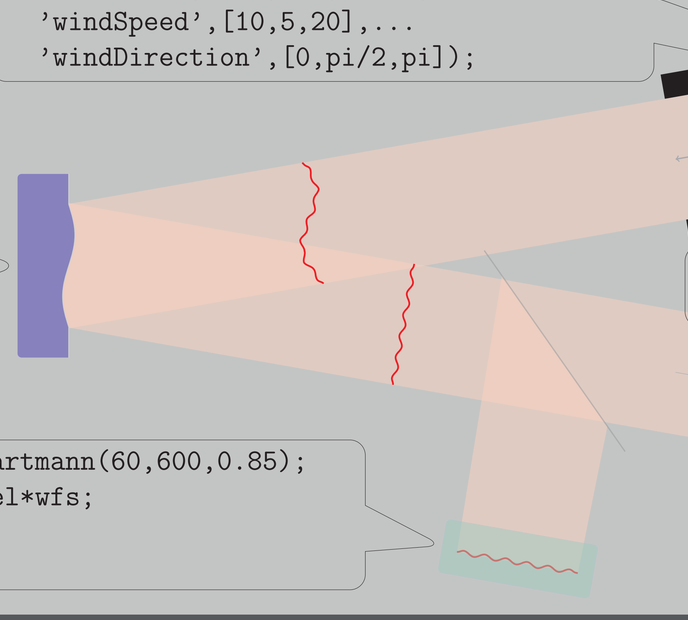
Propagation of the sources:

```
tel = tel + atm;
ngs = ngs.*tel*dm*wfs;
dm.coefs = -calibDm.M*wfs.slopes;
science = science.*tel*dm*cam;
```

```
atm = atmosphere(photometry.V,20e-2,30,...
   'fractionnalR0',[0.5,0.3,0.2],...
   'altitude',[0e3,5e3,12e3],...
   'windSpeed',[10,5,20],...
   'windDirection',[0,pi/2,pi]);
```

```
science = source('wavelength',photometry.J);
```

```
bif = influenceFunction('monotonic',0.75);
dm = deformableMirror(61,'modes',bif,...
   'resolution',tel.resolution,...
   'validActuator',wfs.validActuator);
ngs = ngs.*tel;
calibDm = calibration(dm,wfs,ngs,...
   ngs.wavelength,nL+1,'cond',1e2);
```




```
ngs =source('zenith',arcsec(20));
```

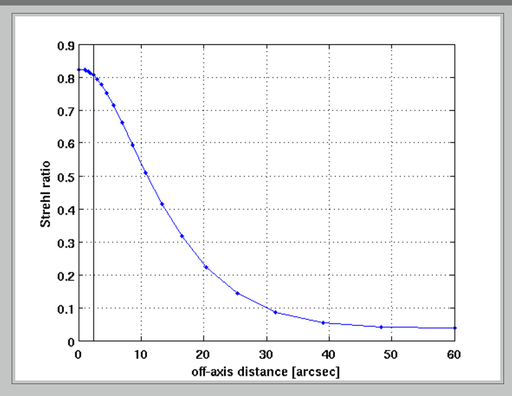
```
tel = telescope(25,'resolution',600,...
   'fieldOfViewInArcsec',30,'samplingTime',1/500);
```

```
cam = imager(tel);
science = science.*tel*cam;
cam.referenceFrame = cam.frame;
```

```
wfs = shackHartmann(60,600,0.85);
ngs = ngs.*tel*wfs;
wfs.INIT
+wfs;
```

Image Plane Sampling

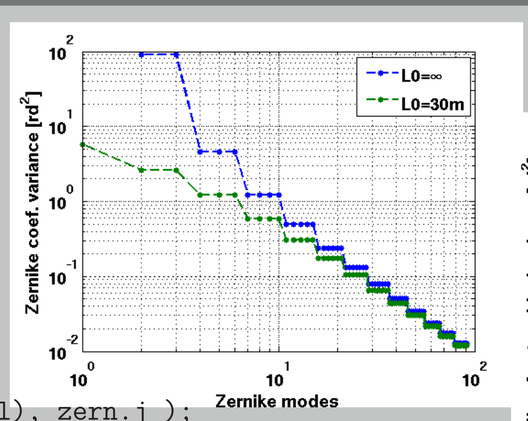
```
science = source('zenith',arcsec([0,logspace(0,log10(60),20)]),'azimuth',zeros(1,21),'wavelength',photometry.H);
source*tel*wfs;
dm.coefs = -calibDm.M*wfs.slopes;
science = science.*tel*dm*cam; figure, imagesc(cam)
figure, plot([science.zenith]*constants.radian2arcsec,cam.strehl,'.-')
```



Analytics

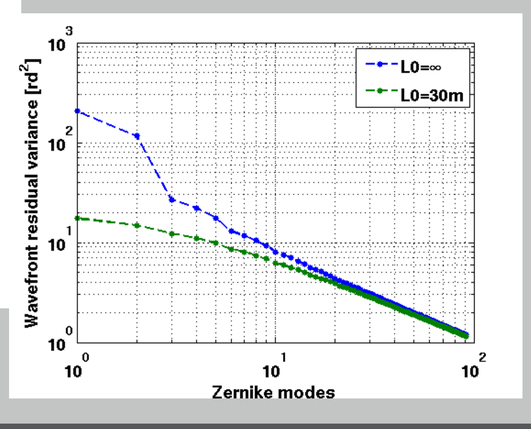
Zernike variance:

```
zern = zernike(tel,...
   1:zernike.nModeFromRadialOrder(12));
atm.L0 = 30;
zvVK = zernikeStats.variance(zern,atm);
atm.L0 = Inf;
zvK = zernikeStats.variance(zern,atm);
```



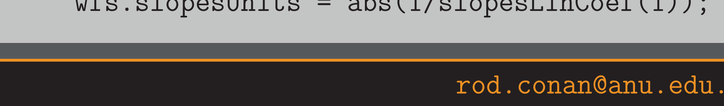
Residual variance:

```
atm.L0 = 30; zveVK = arrayfun( @(x) ...
   zernikeStats.residualVariance(x,atm,tel), zern.j );
atm.L0 = Inf; zveK = arrayfun( @(x) ...
   zernikeStats.residualVariance(x,atm,tel), zern.j );
```



Sodium Laser Guide Star Wavefront Sensing

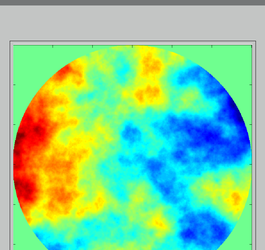
```
nLgs = 21; u = linspace(-1,1,nLgs);
p=exp(-((u-0.35)*5).^2)+0.65*exp(-((u+0.45)*4).^2);
lgs = laserGuideStar(25/60,25,90e3,1,1e9,p,...
    'wavelength',photometry.Na,...
    'height',1e3*(linspace(-5,5,nLgs)+90),...
    'viewPoint',[-25/2,0]);
tel = telescope(25,'resolution',60*48);
wfs = shackHartmann(60,60*16);
wfs.lenslets.fieldStopSize = 24;
lgs = lgs.*tel*wfs; imagesc(wfs.camera)
```



Open–Loop Laser Tomography Adaptive Optics

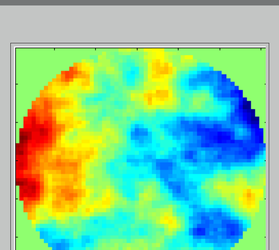
1. Science wavefront:

```
science = science.*tel;
figure(11)
imagesc(...
   science.meanRmOpd*1e6)
axis square
```



2. Science wavefront estimation:

```
lgsAst = source('asterism',{[6,arcsec(25),0]},...
   'wavelength',photometry.Na,'height',90e3);
slmmse = slopesLinearMMSE(wfs,tel,atm,lgsAst,...
   'mmseStar',science);
lgsAst = lgsAst.*tel*wfs;
ps_e = slmmse* wfs.slopes;
figure, imagesc(ps_e), axis square
```



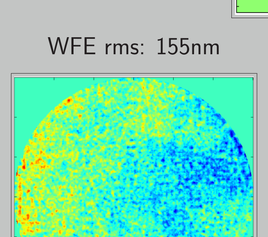
3. DM projection:

```
bifaLowRes = influenceFunction('monotonic',0.5);
dmLowRes = deformableMirror(61,...
    'modes',bifaLowRes,'resolution',61,...
    'validActuator',wfs.validActuator);
F = 2*bifaLowRes.modes(wfs.validActuator,:);
iF = pinv(full(F),1e-1);
dm.coefs = iF*ps_e(dm.validActuator);
```

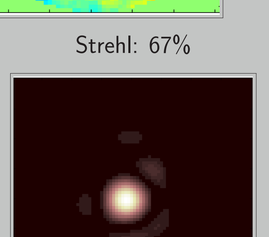
4. Residual wavefront & PSF:

```
science = science*dm*cam;
figure(13)
imagesc(science.meanRmOpd*1e9)
axis square
imagesc(cam)
title(...
   sprintf('Strehl: %2.0f%%',cam.strehl*1e2))
```

WFE rms: 155nm



Strehl: 67%



WFS calibration:

```
wfs.pointingDirection = zeros(2,1);
d = tel.D/wfs.lenslets.nLenslet;
pixelScale = lgsAst(1).wavelength/...
    (2*d*wfs.lenslets.nyquistSampling);
nStep=floor(wfs.lenslets.nLensletImagePx/3)*2;
u = (0:nStep)*pixelScale/2;
u = [-fliplr(u) u(2:end) ];
ngs = source('zenith',u,...
   'azimuth',zeros(1,length(u)),...
    'wavelength',photometry.Na);
tel = tel - atm;
ngs = ngs.*tel*wfs;
u_wfs = median(wfs.slopes(1:end/2,:));
slopesLinCoef =polyfit(u,u_wfs.*pixelScale,1);
wfs.pointingDirection = [];
tel = tel + atm;
wfs.slopesUnits = abs(1/slopesLinCoef(1));
```