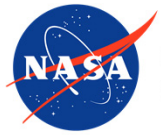


Jet Propulsion Laboratory
California Institute of Technology

Roman CGI Diffraction Model

John Krist
Jet Propulsion Laboratory
California Inst. of Technology
27 July 2021

Public CGI Modeling Software

- PROPER
  - optical propagation library upon which the model is based
  - available for IDL, Matlab, & Python
- roman_phasec_proper
  - CGI Phase C PROPER-based diffraction model
  - Includes telescope & CGI
  - available for IDL, Matlab, & Python
- CGISim
  - Python wrapper around roman_phasec_proper Python model
  - produces intensity images, optionally with EMCCD noise

No WFC Included

None of these packages does wavefront control.

DM patterns for the baseline modes (HLC Band 1, SPC Spec Band 3, SPC WFOV Band 4) are provided in the examples subdirectory of the roman_phasec_proper package for 3 different contrast levels.

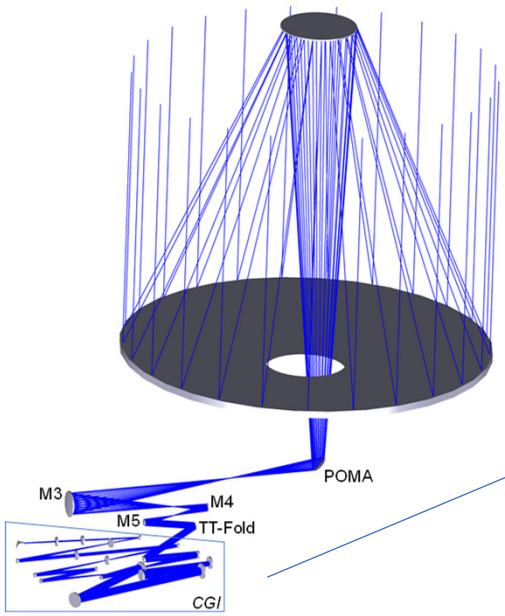
Otherwise, the user must do their own WFC, including creating code to compute the Jacobian. Consider FALCO at https://github.com/ajeldorado

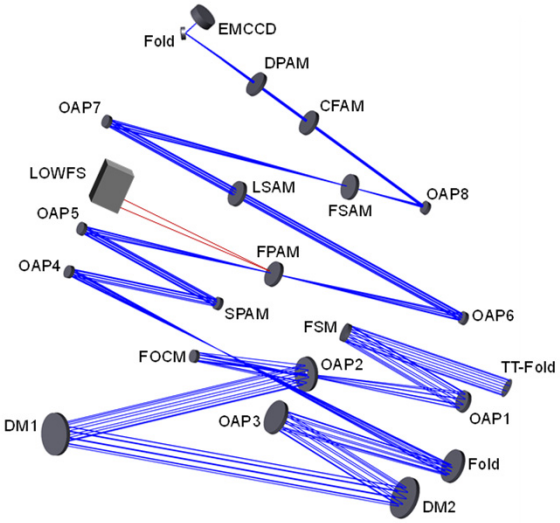
PROPER

- Knowledge of PROPER is not necessary to run the CGI model, except for how to call a PROPER-based prescription (prop_run & prop_run_multi)
  - calls are demonstrated in the model's manual and example programs
- PROPER does the propagating, DM modeling, adding aberrations
- Allows parallel runs (e.g., multiple wavelengths)
- Available from *proper-library.sourceforge.net*

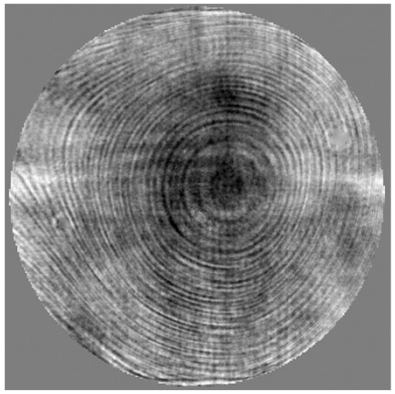
roman_phasec_proper

- Same model is used to generate time series
  - measured primary & secondary mirror errors are used in JPL-only version (ITAR & L3Harris proprietary), while synthetic errors used in public version
- Contains representations of each optic and coronagraphic mask
  - Each optic has either measured or synthetic surface errors
  - As-designed masks used (no mask errors, defaults to perfectly aligned masks)
- Pupil defocus included
- Predicted polarization-induced aberrations included
  - must run model separately for each of 4 polarization terms, resulting images added incoherently
- Options to displace some optics & masks
- User provides DM actuator settings
- Can offset source
- Produces E-field at detector plane for given wavelength
  - HLC FPMs defined at specific wavelengths
- User must introduce jitter by computing multiple source offsets
- A “compact” model is provided, but only as a guide for creating a Jacobian-generating model
- Documentation and multiple examples provided
  - DM settings provided for various contrast levels for baseline CGI modes
- Available from *cgisim.sourceforge.io*

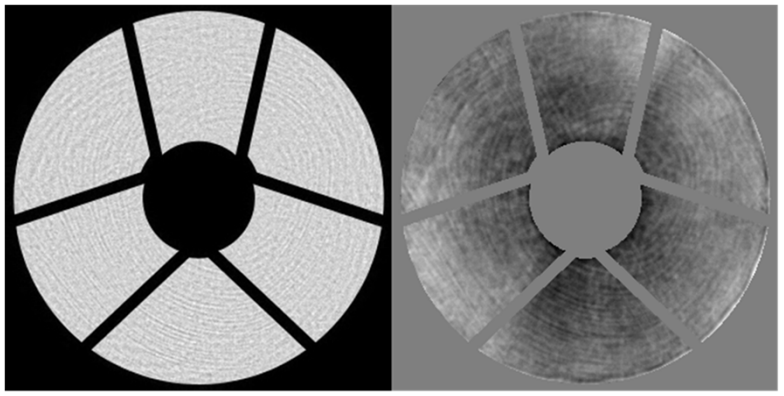




M3 errors

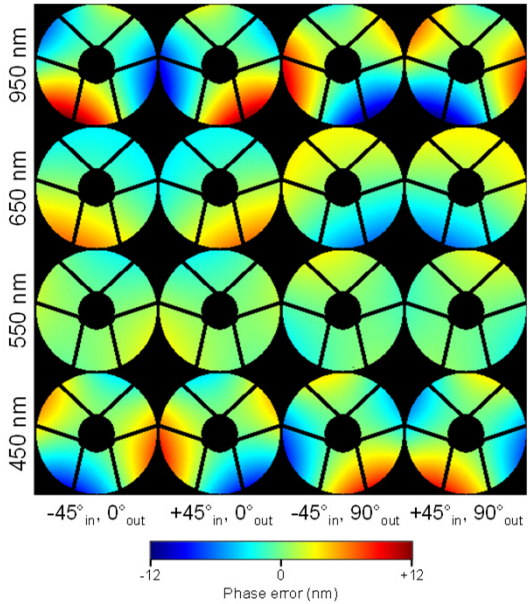


CGI entrance pupil

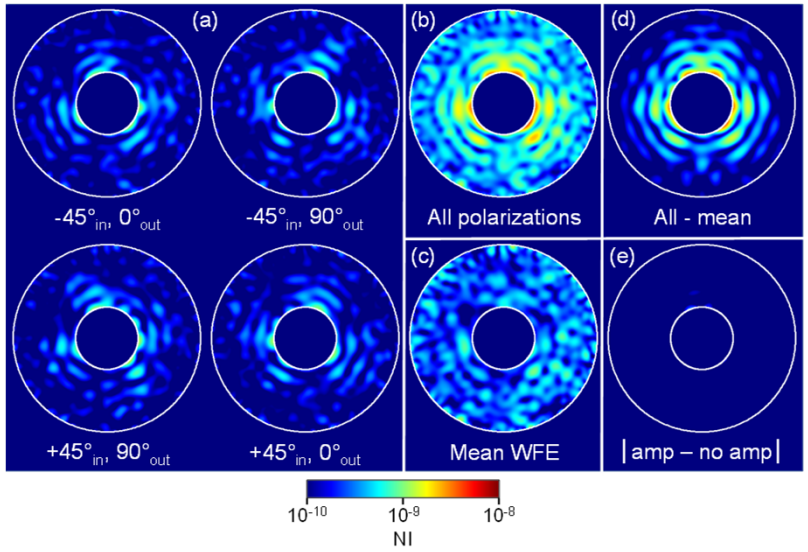


Amplitude Phase

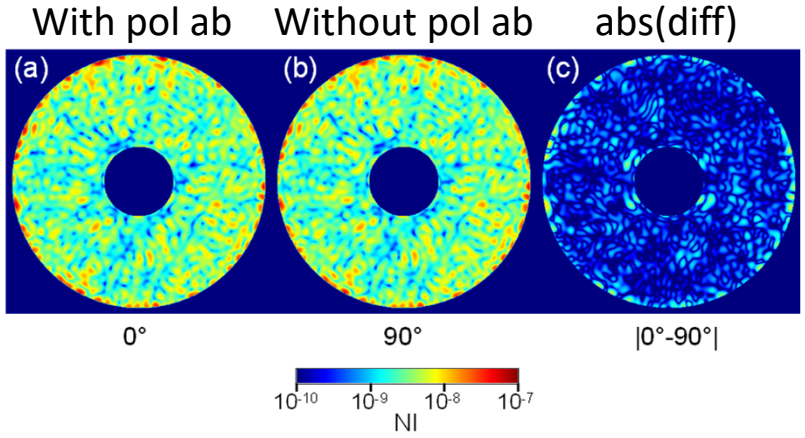
Polarization aberrations



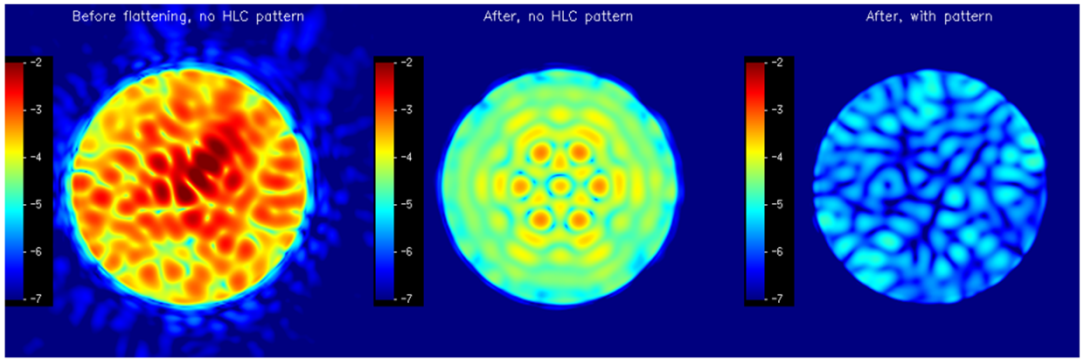
HLC Band 1 models: polarization



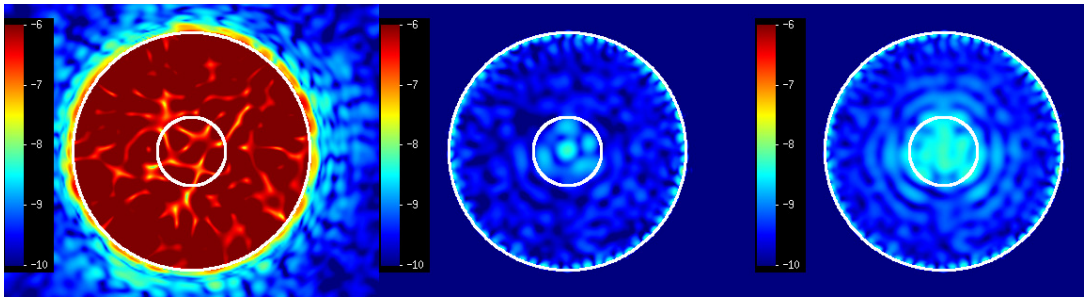
SPC WFOV Band 4 models: polarization



Output of run_flatten.pro



Output of run_hlc.pro



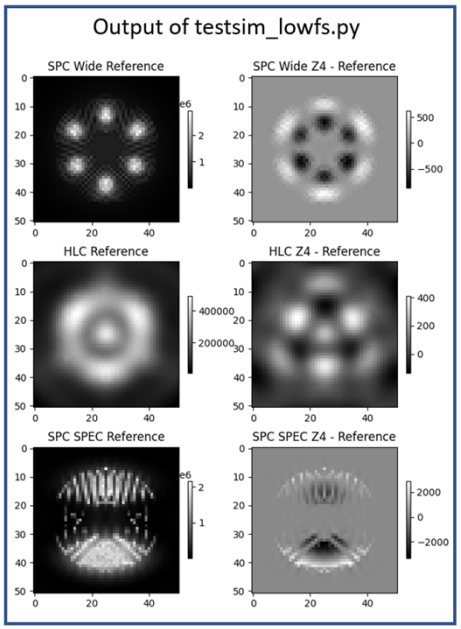
Before EFC

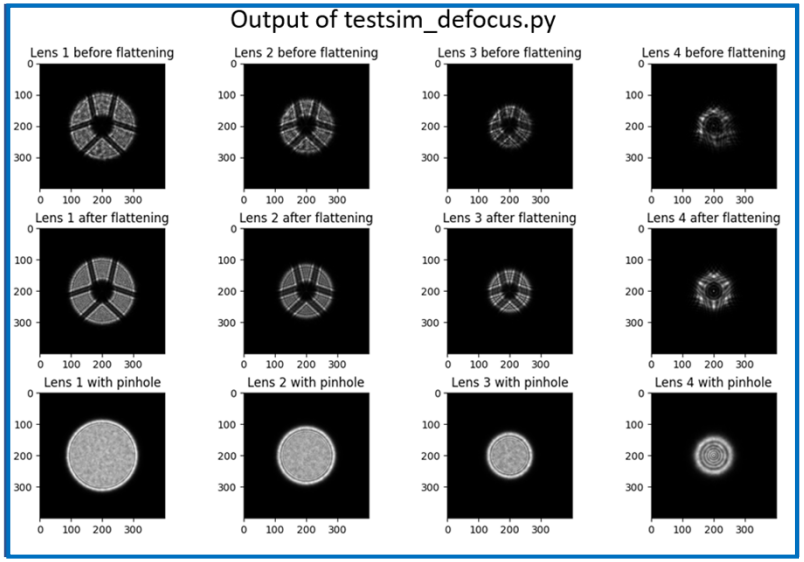
After EFC
mean polarization

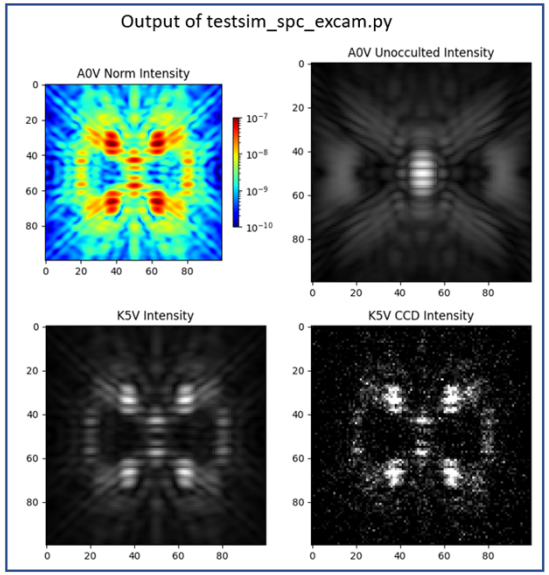
After EFC
full polarization

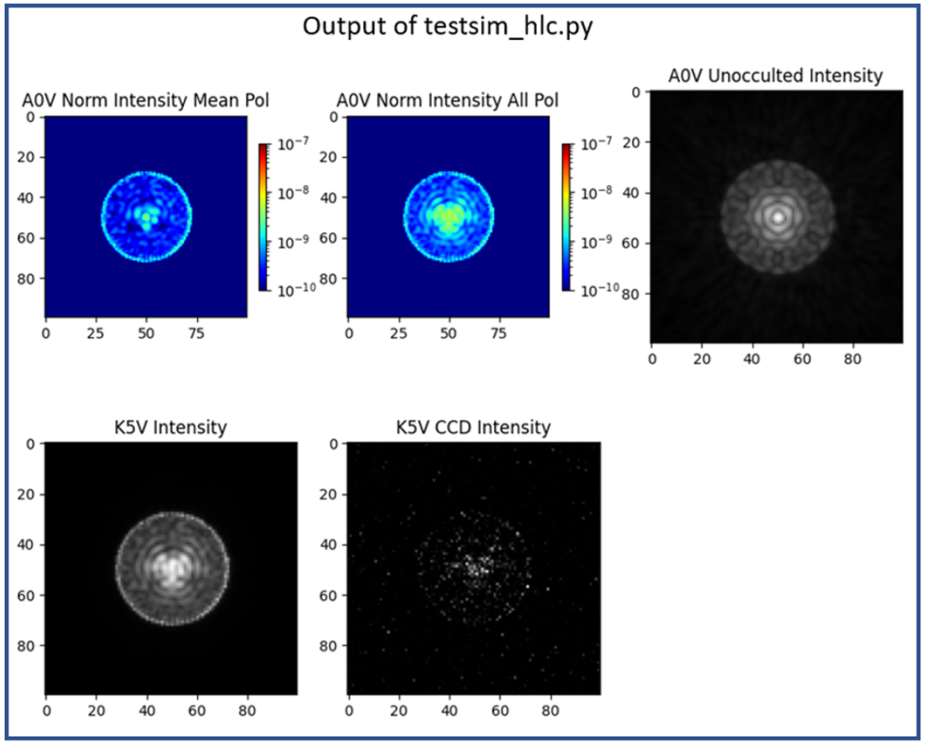
CGISim

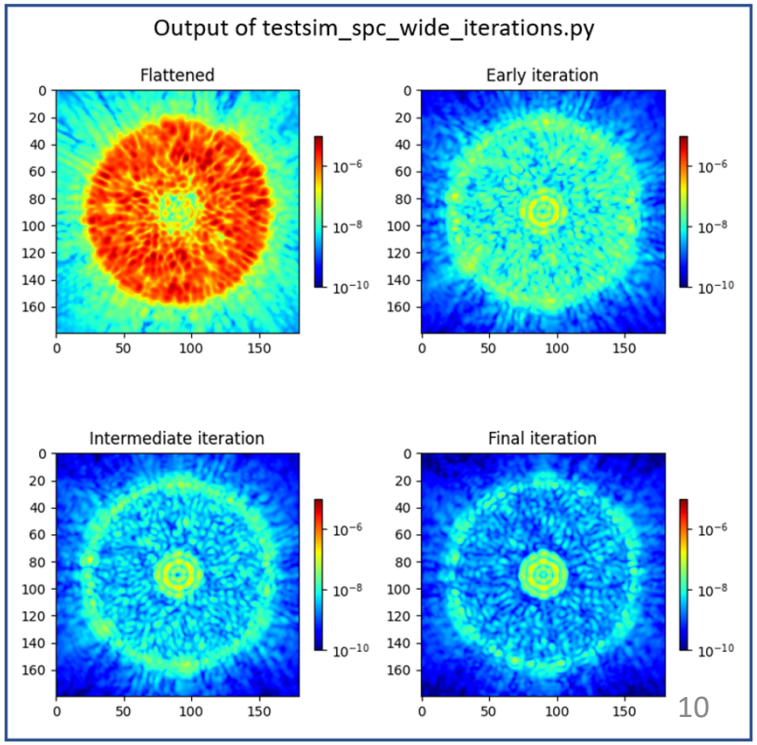
- Wrapper around PROPER model
- User specifies coronagraphic mode, bandpass, DM patterns, stellar brightness & spectrum (limited catalog), exposure parameters
- Optional parameters can be passed to CGI model
- Can produce LOWFS & phase retrieval images
- Produces broadband intensity image at detector-sized pixels, or datacube of images vs wavelength for SPC Spec
- Optionally add EMCCD noise using Nemati & Miller EMCCD_DETECT package
  - does not do the post-processing (photon counting)
  - 1st order solution given in back of manual
  - Nemati describes 3rd order solution in Proc. SPIE, 11443 (2020)
- Primarily has been used for testing wavefront control (phase retrieval) algorithms
  - **Does not** have ability to accept astronomical scenes or add jitter
- Available from *cgisim.sourceforge.io*

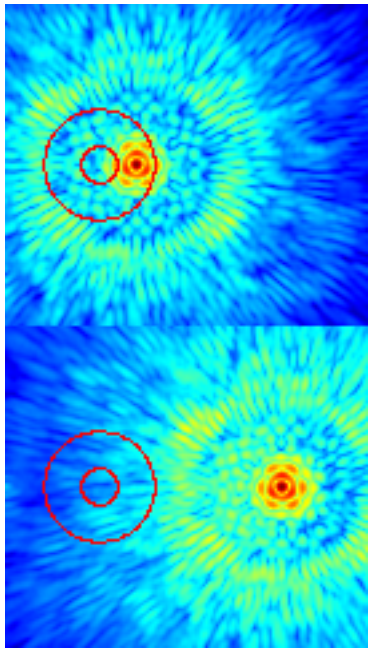

Output of testsim_lowfs.py
SPC Wide Reference
SPC Wide Z4 - Reference
HLC Reference
HLC Z4 - Reference
SPC SPEC Reference
SPC SPEC Z4 - Reference


Output of testsim_defocus.py
Lens 1 before flattening
Lens 2 before flattening
Lens 3 before flattening
Lens 4 before flattening
Lens 1 after flattening
Lens 2 after flattening
Lens 3 after flattening
Lens 4 after flattening
Lens 1 with pinhole
Lens 2 with pinhole
Lens 3 with pinhole
Lens 4 with pinhole


Output of testsim_spc_excam.py
A0V Norm Intensity
A0V Unocculted Intensity
K5V Intensity
K5V CCD Intensity


Output of testsim_hlc.py
A0V Norm Intensity Mean Pol
A0V Norm Intensity All Pol
A0V Unocculted Intensity
K5V Intensity
K5V CCD Intensity


Output of testsim_spc_wide_iterations.py
Flattened
Early iteration
Intermediate iteration
Final iteration




```
import numpy as np
import proper
import roman_phasec_proper
import cgisim


cgi_mode = 'excam'
cor_type = 'hlc'
bandpass = '1'
polaxis = -10       # compute images for mean X+Y polarization

dm1 = proper.prop_fits_read( roman_phasec_proper.lib_dir+'/examples/hlc_best_contrast_dm1.fits' )
dm2 = proper.prop_fits_read( roman_phasec_proper.lib_dir+'/examples/hlc_best_contrast_dm2.fits' )

# source offset by 6 lam/D

params = {'use_dm1':1, 'dm1_m':dm1, 'use_dm2':1, 'dm2_m':dm2, 'use_field_stop':0, 'source_x_offset':6.0}

psf6, counts6 = cgisim.rcgisim( cgi_mode, cor_type, bandpass, polaxis, params,
    star_spectrum='a0v', star_vmag=2.0, output_file='psf6.fits' )

# source offset by 30 lam/D

params = {'use_dm1':1, 'dm1_m':dm1, 'use_dm2':1, 'dm2_m':dm2, 'use_field_stop':0, 'source_x_offset’:30.0}

psf40, counts40 = cgisim.rcgisim( cgi_mode, cor_type, bandpass, polaxis, params,
    star_spectrum='a0v', star_vmag=2.0, output_file= 'psf30.fits' )
```

CGISim Runtimes

- CGISim will run the model using up to 15 threads in parallel (change code to adjust max threads)
- Timings on dual Xeon Gold 6240 workstation (36 real cores)
  - HLC Band 1
    - mean polarization (coherent only, 7 fields) = 0.4 min
    - all polarizations (incoherent, 28 fields) = 1.2 min
  - SPC WFOV Band 4
    - mean polarization (7 fields) = 1.1 min
    - all polarizations (28 fields) = 2.6 min