CGISim (Roman coronagraph) Simulator
Public version (uses roman_preflight_proper with synthetic primary & secondary optic error maps)

V4.0, 27 September 2024
John Krist
Jet Propulsion Laboratory, California Institute of Technology

CGISim is a wrapper around the PROPER Roman CGI prescription. It has most of the parameters pre-defined that are necessary to generate coronagraphic fields, such as bandpass wavelengths, component transmission curves, stellar spectra, and detector pixel sampling. Given a specified coronagraph (e.g., hlc, spc-wide, spc-spec_band2, spc-spec_band3), mode (excam, spec, lowfs), stellar spectral type (e.g., a0v, k5v), and stellar V magnitude, it will generate a coronagraphic image properly sampled for the detector and with appropriate fluxes. CGISim can also call the EMCCD modeling code, emccd_detect, to add detector noise. In "spec" mode a datacube (image vs wavelength) is generated. Nearly all of the PROPER CGI parameters can be used (except those already defined by CGISim); a table is provided later of the prohibited ones. The version of CGISim specific to Roman CGI is **rcgisim**.

NOTE: Because of the large grid sizes needed for the SPC masks, it takes about 2 minutes to compute a broadband image for a single polarization component (there are four such components needed to compute a full polarization image).

Major Changes in v4.0: The Roman CGI model has introduced some major changes:

A new DM model incorporates measured flight DM characteristics, and actuators are now commanded in volts rather than meters of stroke.

The output orientation of the image has changed to match that of the flight instrument. This involves a transpose and rotation compared to what previous versions of the model produced. Note that the actual flight detector orientation is about 0.5° off from nominal X,Y; I have chosen not to include this in order to reduce confusion when the user specifies source and mask offsets.

Charge transfer errors (traps) are no longer an option when including CCD noise effects. This is because I have been unable to successfully build the arcticpy package that models these.

Installing in Python

NOTE: It is strongly recommended that the Intel-optimized numpy and scipy libraries be installed to provide optimal performance. For installation advice, please see the relevant section of the PROPER manual.

Python version 3.x and PROPER version 3.3 or later are required.

Prior to installation of cgisim, other packages should be installed first, in the following order:

1) numpy, scipy, & astropy
2) matplotlib (used by the examples)
3) proper v3.3 or higher (from https://proper-library.sourceforge.net)
4) roman_preflight_proper v2.0 or higher (from
5) emccd_detect v2.4 or higher (developed by Bijan Nemati & Sam Miller at U. Alabama Huntsville), available from https://github.com/roman-corgi/emccd_detect.

The PROPER documentation explains how to install it.

The *cgisim* package is distributed as a zip file. Uncompress it, then enter the top level directory of the distribution.

Enter the directory that contains setup.py and run the following:

```
python -m pip install .
```

When you want to use it, you will need to **`import cgisim`**.

ExCam coronagraph types & bandpasses

cor_type	Allowed Bandpasses
`hlc` *or* `hlc_band1`	1, 1a, 1b, 1c, 1_all
`hlc_band2`	2, 2a, 2b, 2c, 3a, 3b
`hlc_band3`	3, 3a, 3b, 3c, 3d, 3e, 3g
`hlc_band4`	4, 4a, 4b, 4c
`spc-spec_band2`	2, 2a, 2b, 2c, 3a, 3b
`spc-spec_band2_rotated`	2, 2a, 2b, 2c, 3a, 3b
`spc-spec` *or* `spc-spec_band3`	3, 3a, 3b, 3c, 3d, 3e, 3g
`spc-spec_rotated` *or* `spec-spec_band3_rotated`	3, 3a, 3b, 3c, 3d, 3e, 3g
`spc-wide_band1`	1, 1a, 1b, 1c
`spc-wide` *or* `spc-wide_band4`	4, 4a, 4b, 4c
`spc-mswc_band1`	1, 1a, 1b, 1c
`spc-mswc` *or* `spc-mswc_band4`	4, 4a, 4b, 4c
`zwfs`	1, 1a, 1b, 1c, 1_all

Note: Bandpasses 1 & 4 are represented with 7 wavelengths. Bandpasses 3 & 4 use 11 wavelengths. The narrow filters use 5 wavelengths. Band 2 engineering filters include 3a and 3b. *1_all* uses 13 instead of 7 wavelengths to sample Band 1.

CGISim bandpasses

Bandpass Name	Central Wavelength λ_c	FWHM Bandwidth $\Delta\lambda/\lambda_c$
1	575 nm	10.1 %
1a	556 nm	3.5 %
1b	575 nm	3.3 %
1c	594 nm	3.2 %
2	660 nm	17.0 %
2a	615 nm	3.6 %
2b	638 nm	2.8 %
2c	656 nm	1.0 %
3	730 nm	16.7 %
3a	681 nm	3.5 %
3b	704 nm	3.4 %
3c	727 nm	2.8 %
3g	752 nm	3.4 %
3d	754 nm	1.0 %
3e	778 nm	3.5 %
4	825 nm	11.4 %
4a	792 nm	3.5 %
4b	825 nm	3.6 %
4c	857 nm	3.5 %
lowfs	575 nm	22.5%

Calling rcgisim

The usual calling sequence is, with optional parameters in []:

import cgisim
im, counts = **cgisim.rcgisim**(cgi_mode, cor_type, bandpass, polaxis, [, param_struct] [, star_spectrum = *string*]
[, star_vmag = *float*] [, nd = *string*] [, input_save_file = *string*] [, output_save_file = *string*]
[, output_file = *string*] [, ccd = *{dict}*] [, no_integrate_pixels = *True*|*False*]

cgi_mode *(string)*: **'excam'**, **'spec', 'lowfs', 'excam_efield'**

This sets the image sampling, size, and output format.

'excam' is for direct imaging. The available coronagraphs and matching bandpasses are listed in the preceding tables:

The images will be at detector sampling. *counts* contains the returned total number of photons/second from the source of specified spectral type and brightness in the bandpass, excluding any coronagraphic masks but including losses from the OTA obscurations, reflections, filters, and CCD QE. The output image will be 201 x 201 pixels, unless the *fpm_array*, *field_stop_array*, or any phase retrieval modes are defined in the *param_struct* parameter ('use_pupil_lens' or 'use_defocus_lens'), in which case the output will be 511 x 511 pixels.

'spec' is for the spectral mode using the **'spc-spec_band2'** or **'spc-spec_band3'** coronagraphs. The returned array *im* is of dimensions [30,n,n], one image for each of 30 wavelengths uniformly spanning the bandpass. The images will be sampled by 0.1 λ_c/D (λ_c = 660 nm or 730 nm for Band 2 and Band 3, respectively). *counts* is a vector of dimension [30] containing the total number of photons/second from the source of specified spectral type and brightness in each bandpass, excluding any coronagraphic masks but including losses from the OTA obscurations, reflections, filters, and CCD QE.

'lowfs' is for the LOWFS camera and can used with any coronagraph. *counts* contains the total number of photons/second from the source of the specified spectral type and brightness in the bandpass, excluding any coronagraphic masks but including losses from the OTA obscurations, reflections, filters, and CCD QE. The output is a 51 x 51 pixel image at the detector sampling. For the HLC, the reflected field off the FPM is computed using the coating pattern and materials. For the SPCs, the LOWFS dimple is actually a 76 nm raised zone: for spc-wide, a circular spot of r = 0.65 λ/D and for spc-spec_* an elliptical spot of r = 1 x 1.75 λ/D (λ = 575 nm for both), superposed on the reflective surface of the SPC FPM. The only valid bandpass for the LOWFS is 'lowfs'. The geometric pupil diameter is 38.0 pixels on the LOWFS detector.

'excam_efield' is like 'excam", but instead of returning the intensity image in the specified bandpass, it returns the electric field at each wavelength that samples that bandpass. This mode can only be used with no polarization (polaxis = 0) or mean polarization (polaxis = -10). CCD parameters must not be specified, and stellar spectra and flux parameters are ignored. The E-field is returned as a complex-valued variable, normalized to unity flux at the primary at each wavelength. If the *output_file* parameter is specified, it must be the root name (not full name) of the output file – the suffixes *_real.fits* and *_imag.fits* will be added, and the output files will be data cubes.

cor_type *(string)*: **(see table above)**

Defines which coronagraphic masks to use.

bandpass *(string)*: **(see table above)**

Defines the bandpass, which determines how many and which wavelengths to use. Each mode and coronagraph can use only certain filters.

polaxis *(integer)*: **0, 5, 6, 10, -10**

Defines which polarizer, if any, to use (ignored for LOWFS). If polaxis = 0, then no polarization errors are included nor is a polarizer. If polaxis = 5 (X polarizer) or 6 (Y polarizer) then polarization aberrations are included and separate images are generated for the +45° and -45° input polarizations and added together in intensity (a polarizer transmission of 45% is applied). If polaxis = 10 (all polarizations), the four separate images are generated for the four combinations of -45°/+45° in, X/Y out. If polaxis = -10, then the mean of the X and Y polarization aberrations is used and a single image is generated

(this does not include incoherent polarization contributions). The table below describes these modes. **For the most realistic results, you should use polaxis = 10.**

***polaxis* parameters**

polaxis	Polarization states included	Number of incoherent images generated to create final image (per λ)	Polarizer
0	none	1	none
5	$-45°_{in}$, X_{out} & $+45°_{in}$, X_{out}	2	X (45% transmission)
6	$-45°_{in}$, Y_{out} & $+45°_{in}$, Y_{out}	2	Y (45% transmission)
10	$\pm45°_{in}$, X/Y_{out}	4	none
-10	mean of $\pm45°_{in}$, X/Y_{out}	1	none

param_struct *(dict) (optional)*

Passes parameters to the underlying PROPER prescription as a dict. This includes DM settings, Zernike aberrations, mask shifts, etc. All of the parameters listed in the CGI PROPER model prescription documentation may be given, except the following that are otherwise defined by CGISim or are not applicable:

*Parameters **NOT** allowed to be passed to the PROPER prescription from rcgisim*

cor_type	input_field_rootname
end_at_fpm_exit_pupil	output_field_rootname
end_at_fsm	polaxis
final_sampling_lam0	use_fpm (LOWFS only)
final_sampling_m	

star_spectrum = *(string) (optional)*

Specifies the spectral type of the star being observed. The stellar spectrum is integrated over the specified bandpass and normalized by *star_vmag*. The default is ‘a0v’. Only the following spectral types are supported: 'b3v', ‘a0v’, ‘a5v’, ‘f5v’, ‘g0v’, ‘g5v’, ‘k0v’, ‘k5v’, ‘m0v’, ‘m5v’

star_vmag = *(float) (optional)*

Specifies the V magnitude of the star being observed. The default is 0.0.

nd = *(string) (optional)*

Specifies an optional ND filter, as a string: ‘2.25’, ‘4.75fpam’, ‘4.75fsam’

Combining FPAM and FSAM ND filters is currently not supported.

output_file = *(string) (optional)*

The filename, including extension, of the FITS file to which the resulting images are written. If *cgi_mode* is ‘excam_efield’, then this is instead the root name of the output file, and the suffixes *_real.fits* and *_imag.fits* will be added.

output_save_file = *(string) (optional)*

The filename, including extension, of the FITS file to which the intermedate images are written for later reprocessing (different star type or magnitude, different CCD parameters). The images in this file are NOT normalized to any spectral type or magnitude nor have any CCD noise. This cannot be used with *cgi_mode* = 'excam_efield'.

input_save_file = *(string) (optional)*

The filename, including extension, of the FITS file to which the intermedate images are read for reprocessing (different star type or magnitude, different CCD parameters). This is file specified by *output_save_file*. The mode, coronagraph type, bandpass, and polarization used to generate the save file are used.

no_integrate_pixels = True *or* False *(optional)*

By default, images in the ExCam mode are integrated over detector-sized pixels by upsampling the E-fields by 7x the pixel size, then converting to intensity, and then binning down by 7x. Setting this parameter to False will skip this step and the result will be at pixel spacing but not integrated over the pixel area. NOTE: SPEC mode data cubes are not integrated over pixel areas and are provided at 0.1 λ_c/D sampling. LOWFS data are always integrated over pixels.

ccd = *(dict) (optional)*

A dict of EMCCD imaging parameters; if specified, then Bijan's EMCCD model will be applied to the images (otherwise they will be noiseless). The available parameters are given in the table below. The CCD is applied only in 'excam' mode. LOWFS noise assumes 0.5 msec frame times integrated over exptime. Cannot be specified if *cgi_mode* is 'excam_efield'.

EMCCD parameters

Parameter	Default value	Units	Description
'exptime'	no default	seconds	Frame time
'gain'	1000	e^-/photon	CCD gain
'full_well_serial'	100000	e^-	Serial register capacity
'full_well'	60000	e^-	Readout register capacity
'dark_rate'	0.00056	e^-/pixel/sec	Dark rate
'cic_noise'	0.01	e^-/pixel/frame	Charge injection noise
'read_noise'	100	e^-/pixel/frame	Read noise
'bias'	0	e^-	Bias offset
'cr_rate'	0	hits/cm^2/sec	Cosmic ray rate (5 for L2)
'apply_smear'	True (LOWFS)	True or False	Apply LOWFS readout smear
'e_per_dn'	1	e^- / data number	Electrons per data number
'nbits'	14	bits	Number of analog-to-digital converter bits
'numel_gain_register'	604	elements	Number of gain register elements

Simple call examples

An example of the most basic call is:

```
import cgisim
mode = 'excam'
cor = 'hlc'
bandpass = '1'
star_spectrum = 'a0v'
vmag = 2.0
polaxis = 10

image, counts = cgisim.rcgisim( mode, cor, bandpass, polaxis, star_spectrum=star_spectrum, star_vmag=vmag )
```

This will return in *image* the 10% Band 1 image at detector sampling, having been integrated over detector-sized pixels (by upsizing and then rebinning). The image is in photons per second appropriate for the given spectral type and V magnitude (all losses, including reflections and transmissions, are included). The stellar count rate returned in *counts* is what would be measured at the final image

plane without any losses from anything (no coronagraph, no reflection losses, etc.). The image in this example includes all polarizations (*polaxis* = 10).

A contrived example of a more elaborate call that passes parameters to the roman_preflight PROPER prescription:

```
params = {'use_dm1':1, 'dm1_m':dm1, 'use_dm2':1, 'dm2_m':dm2, 'source_x_offset_mas':3.0}

image, counts = cgisim.rcgisim( 'excam', 'hlc', '1', 10, params, star_spectrum='a0v', star_vmag=2.0 )
```

Here *params* is a list of parameters that are passed directly to the PROPER CGI prescription: use the provided DM settings (48x48 arrays containing DM strokes in meters) and offset the source by 3 mas by adding tilt at the primary. In this case, the HLC design is used.

Generate once, use many times

There may be times when you want to use the same coronagraphic field but with a different spectral type/magnitude or CCD noise. In that case, you can save the intermediate image-vs-wavelength datacube (with *output_save_file*) and then read it in (with *input_save_file*) and apply different settings. The saved intermediate datacube does not have any wavelength dependent weighting (stellar spectrum or brightness, system throughput, or neutral density) applied. For example, we can generate an HLC field with no polarizer (incoherent polarization effects included with *polaxis*=10) using provided DM settings for a G5V V=5.0 star and a 10 sec CCD exposure with a gain of 1000, then reuse those fields to create another image for a F5V star:

```
import roman_preflight_proper as rp
import cgisim

bandpass = '1'
polaxis = 10

dm1 = proper.prop_fits_read( rp.lib_dir+'/dm_pistons/hlc_aberrated_dm1.fits' )
dm2 = proper.prop_fits_read( rp.lib_dir+'/dm_pistons/hlc_aberrated_dm2.fits' )

proper_params = { 'use_dm1':1, 'dm1_m':dm1, 'use_dm2':1, 'dm2_m':dm2 }
ccd_params = { 'exptime':10.0, 'gain':1000 }

# generate images from scratch, add CCD noise

gimage, gcounts = cgisim.rcgisim( 'excam', 'hlc', bandpass, polaxis, proper_params,
    star_spectrum='g5v', star_vmag=5.0, ccd=ccd_params, output_save_file='tempfile.fits' )

# use previous images to create new images with different spectral type and different CCD noise realization

fimage, fcounts = cgisim.rcgisim( input_save_file='tempfile.fits', star_spectrum='f5v', star_vmag=5.0  )
```

Jitter is "left as an exercise for the reader"

This version of CGISim does not explicitly support jitter, which is computationally expensive due to the number of source offsets required. It is possible to indirectly implement it by generating separate images with different source offsets spanning the jitter distribution, saving each intermediate result as a different *output_save_file*. All of those can then be read in, appropriately weighted, and then added together, writing it to a new file (keeping the header info from the save file). This new save file can then be processed using *input_save_file*. Star offsets can be specified using the source_x_offset, source_y_offset, source_x_offset_mas, or source_y_offset_mas optional parameters to the PROPER prescription. Jitter of field sources can be directly implemented by convolving with a Gaussian.

DM Solutions

Dark hole DM solutions for the coronagraphs are provided in the examples directory of the roman_preflight_proper package. See that package's documentation for details.

Examples

Some examples are included in the cgisim distribution. To copy these into your current working directory, do the following inside Python:

```
import cgisim
cgisim.copy_examples_here()
```

The examples can be run like so:

```
python testsim_hlc.py
```

The examples are:

testsim_defocus.py

Generate and display three images in the short-end 3.3% bandpass: pupil imaging lens image, defocused image before flattening, defocused image after flattening. Also show images with a 3 micron pinhole in the FPM plane.

testsim_hlc_iterations.py

Generate an HLC images using predefined DM settings in an aberrated system for the mean X+Y polarization (no incoherent polarization contributions). Images are shown prior to running EFC, and then with early, intermediate, and final EFC solutions.

testsim_hlc.py

Generate an HLC (Dwight's) image using predefined DM settings in an aberrated system for the mean X+Y polarization (no incoherent polarization contributions). Show a noiseless A0V, V=2 image (normalized intensity), an unocculted image, a K5V, V=2 image (intensity), and the K5V image with CCD noise in a 30 sec exposure with gain of 1000.

testsim_spc_spec.py

Generate an SPC spectral mode image-vs-wavelength datacube (30 wavelengths) without any DM corrections for the mean X+Y polarization (no incoherent polarization contributions) for an A0V star, V=2 (noiseless). The result is written to a FITS file.

testsim_spc_excam.py

Generate an ExCam image using the spc-spec_band3 coronagraph using predefined DM settings in an aberrated system for the mean X+Y polarization (no incoherent polarization contributions). Show a noiseless A0V, V=2 image (normalized intensity), an unocculted image, a K5V, V=2 image (intensity), and the K5V image with CCD noise in a 30 sec exposure with gain of 1000.

testsim_spc_wide.py

Generate SPC wide FOV images for an unaberrated system and an aberrated system with no wavefront control, with flattening, and with an EFC-derived solution. An offset source image is also computed to convert intensity to normalized intensity. Normalized intensity images are displayed. The results are written to FITS files.

testsim_mask_shift.py

Generate an HLC (Dwight's) image using predefined DM settings in an aberrated system for the mean X+Y polarization (no incoherent polarization contributions). Compute the unocculted image (for computing normalized intensity). Shift the Lyot stop by 2 microns and compute a new image. Show the image with the unshifted stop and plot its azimuthal mean profile and then the azimuthal RMS profile of its difference with the shifted-Lyot-stop image.

testsim_lowfs.py

Generate LOWFS images with the HLC and SPC wide FOV coronagraphs. For each, images are computed in an undisturbed state (reference image) and with 100 pm RMS of defocus added. The reference image and the difference of the perturbed and reference images are displayed. The results are written to FITS files.

testsim_lowfs_emccd.py

Generate a LOWFS image with the HLC for a V=2 A0V star. Show the noiseless result, a single 0.5 ms frame with EMCCD noise, and 20 coadded frames (gain = 100).

testsim_zwfs.py

Generate a Zernike wavefront sensor image.

Changes

8 Oct 2020 - v2.0.4 - J. Krist - fixed bug when adding EMCCD to LOWFS image

2 Nov 2020 - v2.0.5 - J. Krist - increased output image size when using fpm_array or field_stop_array options to PROPER model

2 Apr 2021 - v3.0 - J. Krist - added optional coronagraph modes; updated bandpasses; single frame images are now returned rather than broadband and subband images, due to incorporation of specified FWHMs for each filter

17 May 2021 - v3.0.1 - J. Krist - updated call to emccd_detect to use v2.2

29 June 2021 - v3.0.2 - J. Krist - fixed backwards logic for create_big_image

30 June 2021 - v3.0.3 - J. Krist - added hlc_band1 and spc-spec to list of valid modes

2 Aug 2021 - V3.0.4 - J. Krist - edits to manual only

20 Sept 2021 - v3.0.5 - J. Krist - added option to return or write out E-fields instead of intensities (polaxis=0 or -10, only)

7 Oct 2021 - v3.0.6 - J. Krist - added wavefront reference radii to FITS headers (only for 'excam_efield' mode)

16 Nov 2021 - v3.0.7 - J. Krist - added spc-spec_band2_rotated and spc-spec_band3_rotated options; added filters 3a and 3b to valid filter list for spc-spec_band2 and spc-spec_band2_rotated

23 March 2022 – v3.0.7a – J. Krist – fix to manual (replaced incorrect “spectral_type” with correct “star_spectrum”)

17 Oct 2022 – v3.1 – J.Krist – updated filter and EMCCD curves; added support for Zernike WFS; added support for CTI via arcticpy (off by default)

20 March 2023 – v3.2 – J. Krist – added measured ND filters and changed ND filter specifiers

7 August 2024 – v4.0 – J. Krist – fixed deprication errors; switch to using roman_preflight_proper; remove option to include traps