Roman Preflight PROPER Prescription for IDL, Python, and Matlab
Public version (synthetic primary and secondary maps)
V2.0, September 2024
John Krist

For more details on the workings of this model, see Krist et al., "End-to-end numerical modeling of the Roman Space Telescope coronagraph", JATIS (2023); https://doi.org/10.1117/1.JATIS.9.4.045002

IMPORTANT CHANGES: Versions 2.0 and later of the CGI PROPER model, now called the "preflight" rather than "phasec" model, has some significant changes:

- A new deformable mirror model is now used that incorporates the measured flight DM characteristics, including voltage-dependent gains for each actuator, actuator coupling, dead actuators, influence functions, and surface errors. DM strokes are now specified in volts rather than meters of piston, and voltage resolution and stroke limits apply. The tilt axes of the DMs have also changed. See the "Deformable Mirrors" section of this document for more information.
- The output orientation of the images has changed to match that of the actual CGI images. This involves a transpose and rotation compared to what the previous models produced. At focus-masks (FPM, field stop) and image offset parameters match this new orientation; pupil masks continue to use the wavefront orientation convention (see the "User-defined Masks" section). When in doubt, use the pupil image lens to view pupil mask offsets.
- The Python version no longer needs the user to enter the data subdirectory and run set_data_dir.py to add it to the search path; the data directory is now included as part of the Python distribution.
- An new option, small_spc_grid, is available to compute the spc_spec and spc_wide modes using smaller (500 pix across pupil) grids for faster computation (however, they are not compatible with DM solutions generated for the full-resolution pupils).
- See the change log at the end of this document for other, less significant changes.

Introduction

This version of the PROPER model reflects the final CGI layout and coronagraph masks (though the masks are as-designed, without fabrication errors). There are two models, the "full" and "compact" versions: `roman_preflight` and `roman_preflight_compact`. The full version has every optic included with aberrations, defocus and pupil imaging lenses, polarization aberrations, an optional pinhole at the FPM, and the ability to shift masks or the CGI. It uses PROPER to propagate from optic to optic. It also has the pupil image within CGI displaced along the optical axis to represent the worst-case pupil defocus produced by the telescope's TCA optics (the actual pupil is in-focus on a tilted and slightly curved surface). The compact version only includes the entrance pupil, DMs, and masks. It starts at DM1 and uses PROPER to propagate to DM2 and then back to DM1, after which FFTs are used to go from pupils to focal planes (MFTs are used for part of this when propagating to the FPM at high resolution). It does not include the pupil defocus. It only allows a limited subset of the parameters that are available to the full version. It is useful mainly as an example framework for laying out a quick Jacobian computation model.

This model only propagates the wavefront through the system. It does not include wavefront sensing and control algorithms, such as pairwise-probing or electric field conjugation. Example DM solutions that produce dark holes are provided for the base modes, but otherwise the user must provide their own codes that call the model to represent the actual coronagraph. One such code is FALCO (https://github.com/ajeldorado).

Things to Know

Measured errors: The model includes the measured surface errors for the following optics: primary, secondary, CVS OAPs, POMA fold, M3, M4, M5, TT fold, FSM, FCM (EDU), Fold 3, defocus & pupil imaging lenses. The CGI OAPs use synthetic maps based on

measurements of the flight optics (the measurements are currently too noisy to use directly). The predicted polarization-dependent aberration (tilt, astigmatism) are used. Prior to v2.0, an additional low-order WFE was added (when not using CVS) to bring the total WFE entering CGI to close to the budgeted level, but recent measurements indicate that it was not needed. Measured pupil imaging and defocus lens parameters are included.

Additional options are provided to rotate the CVS pupil, SPC pupil mask, and Lyot stops.

A bulk offset of the beam can be introduced at the SPAM using the spam_*shift* parameters.

Pupil defocus: When using the Roman OTA (default), the distance between the TT fold and the FSM is artificially increased by ~33 mm to introduce defocus of the pupil image, so that the pupil is now 6 mm in front of the SPC pupil mask. This defocus carries on throughout CGI by differing amount depending on the local beam properties. This represents the largest amount of defocus present in the non-planar pupil image produced by the TCA. The compact model does not have this pupil defocus; the delta fields it produces are valid to use with the full model, however. Due to an increase in the distance between OAP8 and the lenses, the pupil image is not at the front focus of the lens. This introduces a radially-quadratic phase term at the final image. This term is subtracted out in the full model when the imaging lens is used (it is not subtracted when the pupil imaging or defocus lenses are used, but one typically does not use the complex-valued fields for wavefront sensing in those modes).

Additional non-baseline coronagraph modes are included: hlc_band2, hlc_band3, hlc_band4, spc-mswc (multi-star wavefront control) for bands 1 and 4, and spc-spec_rotated for bands 2 and 3, and the Zernike Wavefront Sensor (ZWFS). DM solutions for them are not provided.

Setting Up

> *Note*: Python users are strongly encouraged to utilize the Intel-optimized versions of numpy and scipy (see the PROPER v3.3 or later manual for instructions on how to install them). IDL versions of PROPER support using FFTW or the Intel math library's FFT, and you are strongly encouraged to install either and activate PROPER's use of one of them, as detailed in the PROPER manual. The IDL version of roman_preflight requires that the Intel math library be installed (versions ≥8.8.1 of IDL use the Intel FFT by default, but I don't have a recent version to test – ffti.pro could be modified to use the default FFT). Matlab already uses an optimized FFT.

Required data files

The optical error maps, polarization aberration tables, and coronagraph mask files are distributed as part of this package in the subdirectory `preflight_data`. IDL and Matlab users will need to manually edit the full (`roman_preflight`) and compact (`roman_preflight_compact`) models to point the variable `data_dir` to this subdirectory.

The data files are automatically installed as part of the Python distribution (no need anymore to run `set_data_dir)`.

Users are strongly encouraged to install the package on a local disk, otherwise accessing the optical error maps and coronagraph mask files over the network may lead to significant slowdowns.

IDL:

PROPER version ≥3.3 must be installed, and it requires the IDL Astronomy User's Library. The PROPER library directory must be added to IDL_PATH using the IDL **`pref_set`** command, as described in the PROPER manual. The model has been tested with IDL v8.5.1.

The user needs to edit the `roman_preflight.pro` and `roman_preflight_compact.pro` files and change the **`data_dir`** variable to point to the `preflight_data` directory that contains the maps and mask files. Also, edit `load_cgi_dm_files.pro` and change the **`dm_files_dir`** to point to the `preflight_data/dm` subdirectory.

The prescriptions can only be executed using PROP_RUN or PROP_RUN_MULTI from within the current directory. The user should copy all of the files from the roman_preflight distribution into their working directory, including the compiled ffti library.

An interface to the Intel Math Kernel Library is provided and is required. The library is available for free from Intel:

https://software.intel.com/content/www/us/en/develop/articles/oneapi-standalone-components.html#onemkl

To compile the interface, do a ".run compile_ffti" in the roman_preflight library directory. You should copy all of the files in that directory to your current working directory.

Python:

PROPER version ≥3.3 and its prerequisite packages must be installed first. It requires Python v3.x (tested with Python 3.12.4).

The Python version is distributed as the **roman_preflight_proper** package. It uses the **numpy**, **scipy**, **astropy**, and **proper** packages, plus, if you want to run the demo scripts, **matplotlib**. To install, unzip the roman_preflight package and go into the python subdirectory (which contains the file setup.py). In there, issue the following command:

```
python -m pip install .        (depending on your setup)
```

To run a prescription, its model code needs to be copied from your local library into your working directory using the **`copy_here()`** command (the model code in the local library is never directly accessed):

```
import roman_preflight_proper as rp
rp.copy_here()
```

NOTE: Whatever versions of the model that you may currently have in your working directory will get overwritten.

NOTE: If you are using the **rcgisim** routine from CGISim, it will copy the full model to the current directory itself.

To execute the prescription using PROP_RUN or PROP_RUN_MULTI, the **`proper`** module must be imported.

Matlab:

PROPER version ≥3.3 and its prerequisite package must be installed first. The preflight_data directory and subdirectories must be put somewhere permanent. The model has been tested with Matlab v2023b.

The user needs to edit the `roman_preflight.m` and `roman_preflight_compact.m` files and change the **`data_dir`** variable to point to the `preflight_data` directory containing the maps and mask files subdirectories. Also, edit `load_cgi_dm_files.m` and change the **`dm_files_dir`** to point to the `preflight_data/dm` subdirectory.

The prescriptions can only be executed using **`prop_run`** or **`prop_run_multi`** from within the current directory. The user should copy all of the .m files from the distribution into their working directory.

General assumptions

This software represents the flight CGI system. The pupil has a X/Y aspect ratio of 0.9909, with the sampling defined along the Y wavefront axis. The DM tilts are each 9.65° (rotation about the X wavefront axis). The actuator spacing is 0.9906 mm. The wavefront is centered at the joint of the four central actuators. The patterns are passed as 48x48 arrays of using the *dm1_v* and *dm2_v* optional parameters (you'll also have to set the *use_dm1* and *use_dm2* parameters). DM2 is offset from DM1 by 0.1 actuator.

CVS (Coronagraph Verification Stimulus, a.k.a. the TVAC source): The default front-end for this model is the Roman OTA + TCA. The CVS can be used instead by setting `use_cvs` = 1. When using CVS, a variety of additional parameters are available:

cvs_source_z_offset_m : Move the CVS point source back and forth to introduce defocus (positive increases the distance to CGI)

cvs_stop_x_shift_m, cvs_stop_y_shift_m, cvs_stop_z_shift_m, cvs_stop_rotation_deg: move the OTA pupil mask relative to the chief ray, and rotate around the optical axis.

cvs_jitter_mirror_x_offset_mas, cvs_jitter_mirror_y_offset_mas -*or*- cvs_jitter_mirror_x_offset, cvs_jitter_mirror_y_offset: induce an apparent offset of the source in mas or λ_c/D using the CVS jitter mirror

Default coronagraphs

Each design has particular hard-coded settings that may change with new designs: available FPM wavelengths (HLC), FPM dimensions (SPC), pupil sampling (e.g., 309 pixels for HLC, 1000 pixels for SPC), computational grid sizes, and bandpass central wavelength (*lambda0_m*). ***Be sure to take note of which wavelengths the HLC FPM is defined at. Use only these wavelengths when running HLC, otherwise it will use the FPM file for the closest wavelength.***

HLC 20190210b*:* This is the HLC design for a 10% bandpass centered at 575 nm. This is the default coronagraph but can also be explicitly selected by setting the *cor_type* parameter to 'hlc' or 'hlc_band1'. It uses an r = 2.8 λ_c/D asymmetric-phase FPM. An r = 9.7 λ_c/D field stop at the back end is included in the full prescription (there is no stop in the compact model). By default, it produces fields at a sampling of 309/1024 = 0.302 λ/D per pixel. The user must use the *final_sampling_lam0* or *final_sampling_m* optional parameters to produce fields at a specific sampling.

The user may only generate fields at wavelengths for which there is a matching complex-valued FPM map. There are FPMs for 13 evenly-spaced wavelengths spanning the 10% band from 0.54625 μm to 0.60375 μm (which also happen to include 7 evenly-spaced wavelengths), which are (μm):

0.54625000 0.55104167 0.55583333 0.56062500 0.56541667 0.57020833 0.57500000
0.57979167 0.58458333 0.58937500 0.59416667 0.59895833 0.60375000

There are also additional 5 wavelengths that span each of the three narrow engineering filters. In all, the available wavelengths are:

0.54625000 0.54627000 0.55104167 0.55113500 0.55583333 0.55600000 0.56062500
0.56086500 0.56541667 0.56551250 0.56573000 0.57020833 0.57025625 0.57500000
0.57974375 0.57979167 0.58448750 0.58449600 0.58458333 0.58924800 0.58937500
0.59400000 0.59416667 0.59875200 0.59895833 0.60350400 0.60375000

Users should run at only these wavelengths; the prescription looks for the closest wavelength to the requested one.

In the *examples* subdirectory of the model distribution are DM settings for the aberrated system; see the "Example DM Settings" section.

HLC 20200617c (Band 2), HLC 20200614b (Band 3), HLC 20200609b (Band 4): Contributed (non-baseline) HLC designs for Bands 2, 3, & 4. These are selected by setting *cor_type* to 'hlc_band2', 'hlc_band3' or 'hlc_band4'. These have r = 2.8 λ_c/D FPMs (λ_c is the center wavelength of Band 2, 3, or 4; see the filter table given later here). There is an r = 9.7 λ_c/D field stop at the back end. The FPMs are defined at specific wavelengths, and only these should be used. The table of wavelengths is provided in the file `fpm_files.txt` located in the corresponding subdirectory of the *preflight_data* directory. By default, it produces fields at a sampling of 309/1024 = 0.302 λ/D per pixel. The user must use the *final_sampling_lam0* or *final_sampling_m* optional parameters to produce fields at a specific sampling.

SPC 20200617 (Spectroscopic mode, including rotated versions)*:* A.J.'s shaped pupil coronagraph for the spectral mode. It has an r = 2.6 – 9.4 λ_c/D bow-tie FPM with a 65° opening angle and is designed for a 15% bandpass. It is selected by setting *cor_type* to 'spc-spec_band2' for the λ_c = 660 nm band and 'spc-spec_band3' for the 730 nm one. The default output sampling is 1000/2048 = 0.488 λ/D per pixel. The user must use the *final_sampling_lam0* or *final_sampling_m* optional parameters to produce fields at a specific sampling. In the *examples* subdirectory of the model distribution are DM settings for the aberrated system; see the "Example DM Settings" section.

Optional SPC modes are 'spc-spec_rotated' (Band 3), 'spc-spec_rotated_band2', and 'spc-spec_rotated_band3' (same as 'spc-spec_rotated'), all of which rotate the bow-tie dark hole field 60°. Note that DM solutions are not provided for these.

The optional 'small_spc_grid' parameter can be set to use a smaller grid size for faster, though slightly less accurate, computation (500 pixel diameter pupil in a 1024 pixel diameter grid; the default is 1000 and 2048 pixels, respectively). Note that a DM solution for the default grid is not compatible with this option due to effects from the resized masks. This option is not available for the rotated modes.

SPC 20200610 (Wide-field mode)*:* A.J.'s shaped pupil coronagraph for wide field of view. It is selected by setting *cor_type* to 'spc-wide' or 'spc-wide_band4' (both for Band 4) or 'spc-wide_band1' for Band 1. Is has a r = 5.6 – 20.4 λ_c/D annular FPM. It operates in the 11.4% (FWHM) Band 4 bandpass centered at 825 nm or the 10% Band 1 centered at 575 nm. The default output sampling is 1000/2048 = 0.488 λ/D per pixel. The user must use the *final_sampling_lam0* or *final_sampling_m* optional parameters to produce fields at a specific sampling.

The optional ‘small_spc_grid’ parameter can be set to use a smaller grid size for faster, though less accurate, computation (500 pixel diameter pupil in a 1024 pixel diameter grid; the default is 1000 and 2048 pixels, respectively). Note that a DM solution for the default grid is not compatible with this option due to effects from the resized masks. This option is not available for the MSWF mode.

In the *examples* subdirectory of the model distribution are DM settings for the aberrated system; see the “Example DM Settings” section.

SPC 20200623 (Multi-star Wide-field mode)*:* A.J.’s shaped pupil coronagraph for wide field of view modified for multi-star wavefront control with a grid of dots in the pupil mask. It is selected by setting *cor_type* to ‘spc-mswc’ or ‘spc-mswc_band4’ (both for Band 4) or ‘spc-mswc_band1’ for Band 1. Is has a r = 5.6 – 20.4 λ_c/D annular FPM. It operates in the 11.4% (FWHM) Band 4 bandpass centered at 825 nm or the 10% Band 1 centered at 575 nm. The default output sampling is 982/2048 = 0.479 λ/D per pixel. The user must use the *final_sampling_lam0* or *final_sampling_m* optional parameters to produce fields at a specific sampling.

Zernike Wavefront Sensor (ZWFS): This is a mask placed at the focal plane with a phase dimple at its center to provide Zernike phase contrast imaging of wavefront errors on the ExCAM, when used with the pupil imaging lens. This is not a LOWFS replacement and is not used with any coronagraph. It is selected by setting *cor_type* to ‘zwfs’. This is used at the same wavelengths as the Band 1 HLC. It is necessary to also explicitly use the pupil imaging lens (*use_pupil_lens* set to 1).

Execution

The Roman prescription is run like any other PROPER prescription, though it has its own set of optional parameters and the grid size parameter is not the same as typically used by other codes.

Calling Sequence

The user passes to `prop_run` or `prop_run_multi` the name of the prescription (`'roman_preflight'` or `'roman_preflight_compact'`), the wavelength (or array of wavelengths) at which fields will be generated, the output grid size in pixels, and any optional parameters as a structure/dict. The prescription returns the E-field and the sampling in meters (which is always 0 if running the compact model).

IDL

The IDL call is:

`prop_run, 'roman_preflight',` *field, lambda_um, gridsize, returned_sampling_m, PASSVALUE*={…}

or

`prop_run_multi, 'roman_preflight',` *fields, lambdas_um, gridsize, returned_samplings_m, PASSVALUE*={…}

Python

The Python call is, after importing the **proper** and **roman_preflight_proper** modules and copying the prescription file to the local directory using `roman_preflight_proper.copy_here()`,:

field, sampling_m = `proper.prop_run( 'roman_preflight',` *lambda_um, gridsize, PASSVALUE*={…})

or

fields, samplings_m = `proper.prop_run_multi( 'roman_preflight_compact',` *lambdas_um, gridsize, PASSVALUE*={…})

Matlab

The Matlab call is (replace *roman_preflight* with the appropriate prescription name):

[*field, sampling_m*] `= prop_run('roman_preflight',` *lambda_um, gridsize, 'passvalue', ...)*

or

[*fields, samplings_m*] `= prop_run_multi('roman_preflight_compact',` *lambdas_um, gridsize, 'passvalue', ...)*

Note that the grid size is not the size of the computational grid, as is commonly assumed by PROPER. Instead, the computational grid size is hard-coded in the prescription and this parameter is instead the dimension of the output result (but only if the output is at the final image plane, otherwise it is ignored). **This value MUST be a power of two, since PROPER assumes it is and checks for it.**

In the case of **`prop_run`**, *lambda_um* is a single value and *PASSVALUE* points to a structure. When using the parallel processing of **`prop_run_multi`**, *lambdas_um* may be an array of wavelengths and/or *PASSVALUE* is an array of structures/dicts. You can either run different sets of parameters for a single wavelength, multiple wavelengths for a single set of parameters, or matched multiple wavelengths and parameters (in which case *lambdas_um* and *PASSVALUE* have same number of array elements).

PASSVALUE is a list of optional keyword:value pairs that override default values. The available parameters are listed in Table 1. An example of the syntax is:

IDL:
```
PASSVALUE={use_errors:1, cor_type:'spc-spec_band3',
           use_dm1:1, dm1_v:dm1_array}
```

Python:
```
PASSVALUE={'use_errors':1, 'cor_type':'spc-spec_band3',
           'use_dm1':1, 'dm1_v':dm1_array}
```

Matlab:
```
optval.use_errors = 1
optval.cor_type = 'spc-spec_band3'
optval.use_dm1 = 1
optval.dm1_v = dm1_array
[field, dx] = prop_run( ……., 'passvalue', optval )
```

Returned Values: Field properties and sampling

The prescription returns the complex-valued electric field and the sampling in meters. Be aware that both depend on which plane the prescription is propagating up to, depending on the parameters set by the user. If **`prop_run`** is used then the field will be a 2-D complex-valued array and *sampling_m* is a scalar. If **`prop_run_multi`** is used, the field will be a 3-D array ([*n, n, nlam*] in IDL and Matlab, [nlam, n, n] in Python) and *sampling_m* will be a vector of corresponding samplings (note here that *n* depends on where the propagation stops and may or may not be *gridsize*, and *nlam* is either the number of wavelengths or the number of PASSVALUE structures passed).

The default behavior is to propagate from the primary (full prescription) or DM1 (compact) through to the final image plane. The sampling there can be set using the *final_sampling_lam0* parameter in units of λ_0/D (λ_0 is the central wavelength of the bandpass, also known here as λ_c); if it is not specified, then the default sampling, which depends on wavelength, is used and so the user should then pay close attention to *sampling_m*. The field at the final image plane will be returned as a *gridsize* by *gridsize* pixel array. There are options to propagate just up to the CGI entrance (the FSM, without FSM aberrations applied) or the FPM exit pupil (actually the Lyot stop plane without aberrations between the FPM and Lyot stop applied). In these cases, the returned arrays ignore the *gridsize* and *final_sampling_lam0* specifications. Instead, the dimensions and sampling are set by the pupil dimensions assumed in the code (the user should inspect the code to see what `pupil_diam_pix` is for the chosen coronagraph).

Example DM Settings

NOTE: The convention for DM commands in the CGI model is actuator voltage.

DM patterns are provided (*_flat_wfe_*.fits) for each baseline coronagraph that produce a flat phase at the FPM in the aberrated system (including mean polarization aberrations). These represent the system after applying correction for the phase-retrieval-measured wavefront error, which is the initial state prior to running EFC.

Dark hole DM patterns are also provided for the aberrated system (including mean polarization aberrations) for each coronagraph. Separate settings are given for good, moderate, and TVAC-level contrasts. Solutions providing the latter two contrasts are from earlier iterations in the same EFC runs that provided the good contrasts. The HLC solutions are for band 1, SPC-Spec for Band 3, and SPC-Wide for Band 4.

The solutions are located in the *examples* subdirectory of the distribution. For installed Python distributions, these can be accessed from:

```
import roman_preflight_proper as rp
examples_directory = rp.lib_dir+'/examples'
```

Provided DM Solutions:

Aberrated system, flat-phase-at-FPM solution (HLC does not have HLC patterns):

```
hlc_flat_wfe_dm*.fits
spc-spec_flat_wfe_dm*.fits
spc-wide_flat_wfe_dm*.fits
```

Best contrast solution in aberrated system:

```
hlc_ni_2e-9_dm*.fits
spc-spec_ni_1e-9_dm*.fits
spc-wide_ni_3e-9_dm*.fits
```

Mild contrast solution in aberrated system:

```
hlc_ni_5e-9_dm*.fits
spc-spec_ni_4e-9_dm*.fits
spc-wide_ni_5e-9_dm*.fits
```

Worst (TVAC-level) contrast solution in aberrated system:

```
hlc_ni_3e-8_dm*.fits
spc-spec_ni_2e-8_dm*.fits
spc-wide_ni_2e-8_dm*.fits
```

Polarization

The full model includes polarization aberrations, but not by default (*polaxis* = 0). To model with the mean aberration, set *polaxis* to 10. To generate a model including the incoherent polarization aberrations (which is what one will typically see), you will need to generate a separate model for each of the four polarization components (polaxis = -2, -1, 1, 2) and average these images ***in intensity***.

Offsets and Shears

Parameters (see Table 1) are provided to offset the source or shift a mask or the entire CGI. Source offsets can be made by introducing tip/tilt at either the primary or FSM. The offsets can be specified either in mas or λ_0/D, and produce the same direction of shift as seen in the final image plane. Pupil mask shifts (SPC pupil mask, Lyot stop) and the CGI as a whole (bulk shift at FSM) are specified as the fraction of the pupil diameter or meters. The FPM can be offset in X and/or Y in units of λ_0/D and along the optical (z) axis in meters.

The focus correction mechanism can be pistoned to adjust focus. Note that this only alters focus; any induced changes in other aberrations have not been accounted for.

Examples

In the examples subdirectory in the model directory (e.g., idl/examples or python/roman_preflight_proper/examples), there are a few example programs. You will need to copy the model into this directory and run it in there.

run_defocus: Compute and display monochromatic (575 nm) images taken with the four defocus lenses and the pupil imaging lens.

run_flatten: Compute SPC-WIDE band 4 images using the full model with 40V on all DM actuators, then with the wavefront-flattened DM solutions. Also compute an offset-source image to get the PSF peak value (used to compute normalized intensity). The images are displayed.

run_hlc: Compute HLC band 1 images using the full model with EFC-derived DM solutions providing a range of mean darkhole contrast, including polarization aberrations. The images are displayed.

run_spc_spec: Compute SPC-SPEC band 3 images using the full model with EFC-derived DM solutions providing a range of mean darkhole contrast, including polarization aberrations. The images are displayed.

run_spc_wide: Compute SPC-WIDE coronagraph band 4 images without aberrations (using the compact model) and with aberrations and flat WFE and EFC-derived DM solutions, for the mean (coherent) polarization aberrations. The images are displayed.

Generating Broadband Images

A broadband simulation is composed of monochromatic intensity images generated at wavelengths sampling the bandpass. In the case of the HLCs, there are specific wavelengths available (corresponding to separate FPM pattern files). There are no restrictions on the SPCs. The CGI bandpasses, as currently defined, are shown below. Use 7 or 13 wavelengths for Bands 1 & 4 and use 9 or more for the broader Bands 2 & 3. Use 3-5 wavelengths for the narrower calibration bandpasses.

Table 1. CGI Bandpasses

Bandpass Name	Central Wavlength 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 %

Using the DM

In versions of the PROPER CGI model prior to 2.0, the PROPER DM model was used directly to represent the CGI DMs. This meant that DM actuator strokes were specified in meters, all actuators responded identically and linearly, there were no intrinsically dead/weak ones, and they all had the same influence function. The actual DMs are quite different: actuators are commanded in volts, each actuator has a unique gain (stroke/V) that varies not-fully-linearly with voltage, voltage resolution is limited by the electronics, voltage range is set to 0 – 100 V, a few actuators are dead or weak, some actuators are electrically coupled to neighbors, and each DM has a unique mean influence function. If you send a uniform voltage to the DM, it will not produce a flat surface because each actuator has a different gain.

Due to desorption in the materials used in the electrical connection to the DMs that deforms the surface, there is some astigmatism (the actual amount of resulting wavefront astigmatism is effectively less due to compensation by selected alignments of the preceding off-axis parabolic mirrors; a residual of 50 nm RMS of Z6 wavefront error is assumed from each DM in the model). These characteristics have been measured for the flight devices using an interferometer in a vacuum tank. Version 2.0 and later of the CGI model uses a new DM model that incorporates all of these. See the Krist et al. (2023) JATIS paper on modeling for a description.

The files used by the DM model are stored in the `preflight_data/dm` directory of the model distribution. DM 1236-2 is DM1 (at a pupil) and DM 1236-5 is DM2 (1 m downstream). See below for a more detailed description of them.

DM Geometry

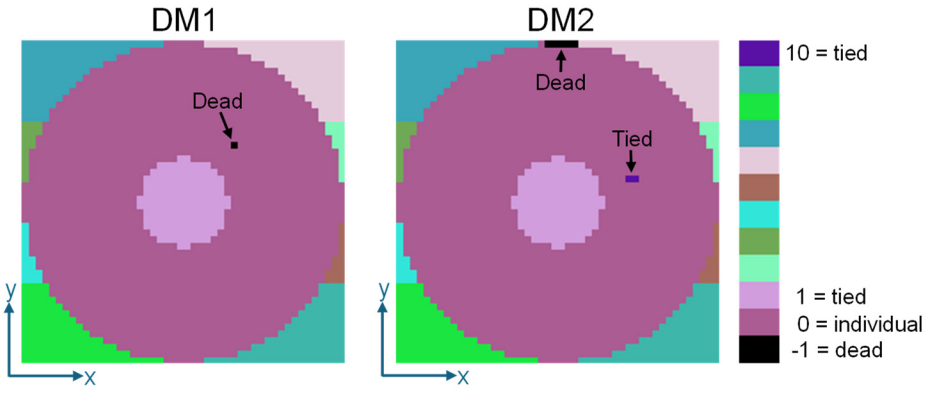
Parameters are provided to alter the default DM actuator spacing (*dm_sampling*), tilt *(dm*_xtilt_deg, dm*_ytilt_deg, dm*_ztilt_deg*), and centration relative to the wavefront center *(dm*xc_act, dm*_yc_act*). The prescription assumes DM tilts of 9.65° (around the local wavefront X axis) and an actuator spacing of 0.9906 mm. TVAC measurements indicate that the center of DM2 is offset relative to DM1 by about 0.1 actuator, so that is included by default. The section “System Geometry” describes the assumed axes orientations – it’s something of a nightmare to contend with, especially if you are trying to create a control model to compute Jacobians.

The DM commanded values (*dm*_v*) are specified as 48 x 48 arrays whose orientation matches the face-on view of each DM. That is, the first array element commands the lower-left corner actuator while looking at the DM surface. Because the DMs face each other, the left side of DM1 is aligned with the right side of DM2, and vice-versa, and the user-commanded values must take this into account.

Active vs. Tied DM Actuators

The files dm*_tiemap.fits are 48 x 48 arrays containing the DM actuator identifier codes. Active, individually addressible actuators have codes of 0, and dead actuators are -1 (both DMs have dead actuators). Then there are tied actuators, which have the same voltage, either because they are intentionally connected to the same driver line (groups with IDs = 1 to 9 on either DM) or there is a complete electrical short between them (ID = 10 on DM2). Acuators that fall within the illumated region of the pupil are in-general individually addressible, while those outside the pupil, or inside the shadow of the central obscuration, are tied together into various groups. Each group of tied actuators have the same code, an integer equal or greater than 1. Note that coupled actuators (described below) are not considered tied, as they do not have equal voltage.

CGI DM Tie Maps with color-coded legend



DM Gains and Voltages

The CGI DMs are electrostrictive, so as applied voltage is increased to an actuator, it pulls the facesheet down towards the DM body. Dead actuators therefore will appear to stick up relative to a DM surface when voltage is sent to all actuators. The voltages are limited to 0 – 100 V. While any voltage within that range can be specified, the digital-to-analog converter provides a resolution of $110/2^{16} \approx$ 1.7 mV, which is included in the model.

The conversion of voltage to stroke is different for each actuator. The measured stroke-vs-voltage function of each is used (rather than a linear approximation, which is 3.6 nm/V for DM1 and 3.6 nm/V for DM2). These functions are stored in the files dm*_stroke_nm_vs_volts_*c_v8.0.fits (there are separate files for the two measurement temperatures, and the values are interpolated to the specified temperature (the default in the model is 26° C).

DM settings are usually derived relative some bias voltage, 40 or 50 V. Because of the gain variations, sending a uniform voltage to the DM will result in a non-flat surface. The settings described in "Example DM Settings" include ones that flatten the system wavefront error prior to the focal plane mask, which includes variations in the DM.

Actuator Coupling

On DM2 there is a region of actuators that are partially electrically connected to adjacent neighbors, mostly in the -Y direction. Unlike tied actuators, which equally share voltage, these actuators partially share their voltage with neighbors. The 3 x 3 coupling distributions for each actuator are stored in the files dm*couplings_*c.fits (the coupling has been measured at two temperatures and multiple voltages and is interpolated to the specified ones). The coupling factors are applied to the strokes, not voltages.

If you are creating a control model to compute Jacobians, it is essential that you include the effect of coupling if you want to create a good dark hole for the SPC wide field of view (or SPC-MSWF), but it is less critical for the HLC and SPC Spec modes.

DM Files

The files used by the DM model are stored in the `preflight_data/dm` directory of the model distribution. DM 1236-2 is DM1 (at a pupil) and DM 1236-5 is DM2 (1 m downstream). The array order is given in IDL convention (fastest varying index first).

`*_brian_inf_v8.0.fits` `[73,73]` `(float)`

Mean actuator influence function for the DM with high sampling (specified in header).

`*_couplings_*c_v8.0.fits` `[3,3,48,48,11]` `(float)`

Fractional coupling, specified as a 3 x 3 array, ordered (x-1,y-1) to (x+1,y+1), for each actuator (x,y). The surrounding actuators receive a fraction of actuator (x,y)'s stroke (not voltage). This is important for DM2, mostly in the -Y direction (DM1 has no significant coupling). The coupling factors are given relative to 11 voltages (0V-100V in 10V increments) and interpolated based on the voltage of actuator (x,y).

`*_stroke_nm_vs_volts_*c_v8.0.fits` `[48,48,11]` `(float)`

The piston (nm) versus volts (0V-100V in 10V increments) for each actuator.

`*_tiemap_v8.0.fits` `[48,48]` `(integer)`

DM actuator identifier codes. Active, individually-addressible actuators have codes of 0, and dead actuators are −1. Tied actuators, which have the same voltage, have the same code, an integer equal or greater than 1.

`*_0v_sfe_m_without_Z1-Z6_v8.0.fits` `[763,763]` `(float)`

Surface error map of polishing errors on the facesheet, measured from the unpowered DM with Z1-Z6 subtracted.

`*_sfe_per_Vbias_v8.0.fits` `[741,741]` `(float)`

Surface error map of residuals after flattening the DM surface, with intra-actuator ripples that vary with bias voltage (meters of surface error per bias voltage).

Using the `load_cgi_dm_files` function

The load_cgi_dm_files function will read in the files described above and populate a structure array that makes access easier. The call is:

IDL: *struct_array* = **load_cgi_dm_files(** dm_files_dir [, VERSION=*string*] [, TEMP_C=*float*] **)**

Matlab: *struct_array* = **load_cgi_dm_files([**'dm_files_dir'**,** dm_files_dir] [, 'version', *string*] [, 'temp_c', *float*] **);**

Python: import roman_preflight as cgi
struct_array = **cgi.load_cgi_dm_files([**dm_files_dir=*string*] [, version=*string*] [, temp_c=*float*] **)**

Parameters in brackets [] are optional. By default, the DM version is '8.0' and the temperature is 26° C. The *dm_files_dir* directory is defined inside the IDL and Matlab codes and can be changed by the user, otherwise specifying the path in the call will override. In Python, the directory is automatically assumed to be a subdirectory in the installed package.

The returned *struct_array* is a two-dimensional array (one for each DM) of structures/lists containing the DM characteristics arrays. The first structure is DM1, the second is DM2. The elements are:

Structure element	type, IDL dimensions	Contains
bias_proportional_sfe_map_m	*float* [741, 741]	Bias-proportion surface error map (meters surface error)
bias_proportional_sfe_map_dx_m	*float*	Sampling of map in meters
coupling	*float* [3, 3, 48, 48, 11]	Actuator coupling coefficients maps (11 voltages)
coupling_volts	*float* [11]	Voltages for coupling map
inf_file	*string*	High resolution mean actuator influence function
static_sfe_map_m	*float* [763, 763]	Surface error in meters
static_sfe_map_dx_m	*float*	Sampling of map in meters
stroke_m_vs_volts	*float* [48, 48, 11]	Stroke versus volts (11 voltages)
stroke_volts	*float* [11]	Voltages for stroke_m_vs_volts
tie_map	*integer* [48,48]	Actuator type ID map
version	*string*	DM files version (e.g., '8.0')

Applying acutator constraints

The CGI project enforces certain constraints on the DM actuators. The allowed voltage range is 0V – 100V. Adjacent actuators cannot differ by more than 50V in the horizontal or vertical directions or more than 75V in diagonal directions to avoid stressing the bonds with the facesheet. The voltages of the violating pairs are adjusted up/down by equal amounts to meet this constraint. Dead actuators are set to 0V and tied actuator voltages are averaged.

A routine to apply these constraints is provided, constrain_dm:

IDL: *float_array* = **constrain_dm(** dm_files_dir [, VERSION=*string*] [, TEMP_C=*float*] **)**

Matlab: *float_array* = **load_cgi_dm_files(** 'dm_files_dir'**,** dm_files_dir [, 'version', *string*] [, 'temp_c', *float*] **);**

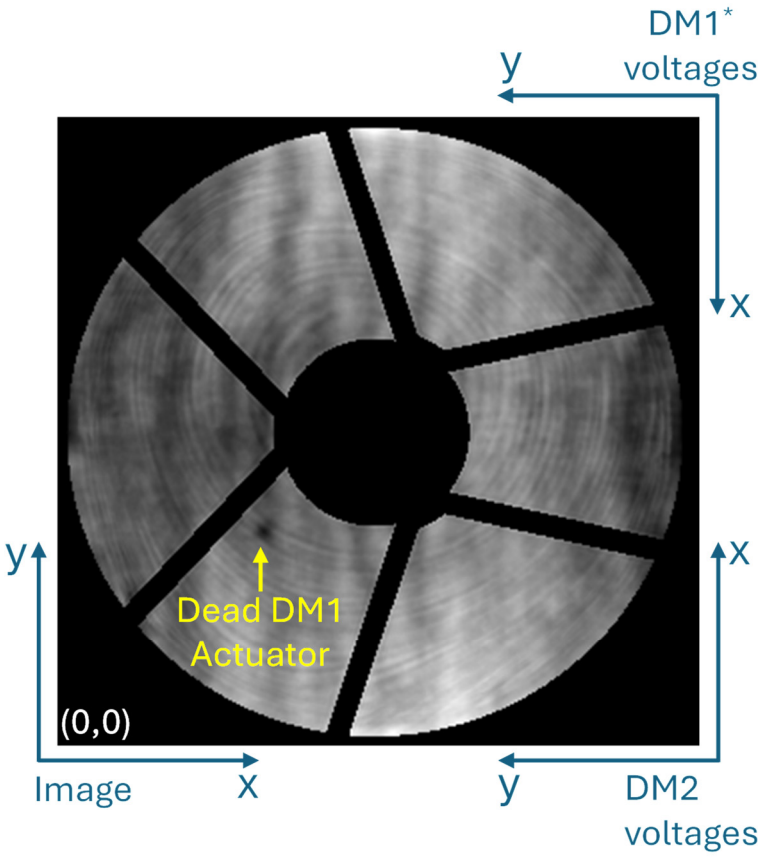
Python: import roman_preflight as cgi
float_array = **cgi.load_cgi_dm_files([**dm_files_dir=*string*] [, version=*string*] [, temp_c=*float*] **)**

Parameters in brackets [] are optional. By default, the DM version is '8.0' and the temperature is 26° C. The dm_files_dir directory is defined inside the IDL and Matlab codes and can be changed by the user, otherwise specifying the path in the call will override. In Python, the directory is automatically assumed to be a subdirectory in the installed package.

The constrain_dm routine runs much slower in Matlab than in IDL or Python, but produces the same results. I don't know why, though Matlab is not my native language. If someone figures out the source of the inefficency, please let me know.

System Geometry

The variety of orientations of axes is confusing. DM voltages are specified in a face-on orientation with axes defined by the electronics. The DM tilts are specified in terms of the local wavefront orientation in the model. The output image orientation is defined by the actual CGI convention. The diagram below shows these orientations.



*DM1 axes are used to define centers (*dm*_xc_act, dm*_yc_act*)
and tilts (*dm*_xtilt_deg, dm*_ytilt_deg*) for both DMs

Phase of the field produced by the model when both DMs are set to a uniform 40V, as computed using the pupil imaging lens (use_pupil_lens = 1). The phase errors are a combination of system surface errors and DM gain variations and astigmatism. The DM input voltage map axes are shown as mapped to the output image.

Table 2. PASSVALUE Parameters

parameters NOT available to rcgisim are marked with (*)

Coronagraph type, data directory, image sampling

Parameter	Valid value or type	Default	Description
cor_type (*)	'hlc' or 'hlc_band1', 'spc-spec_band2', 'spc-spec_band3', 'spc-spec_rotated', 'spc-wide' or 'spc-wide_band4', 'spc-wide_band1', 'spc-mswc_band4', 'spc-mswc_band1', 'hlc_band2', 'hlc_band3', 'hlc_band4', 'zwfs', 'none'	'hlc'	Coronagraph type
data_dir	string	User-specified	Directory containing error maps & mask subdirectories
final_sampling_lam0	λ_0/D units	Varies with coronagraph & wavelength	λ_0 = science bandpass central wavelength (see Table 1)
final_sampling_m	meters	Varies with coronagraph & wavelength	Specify either final_sampling_lam0 or final_sampling_m, but not both
small_spc_grid	0 or 1	0	If 1, uses 500 rather than 1000 pix diameter pupil for SPC (spc-spec or spc-wide only)

Use/exclude CGI components

Parameter	Valid value or type	Default	Description
use_cvs	0 or 1	0	1 = use CVS for the front end rather than Roman
use_aperture	0 or 1	0	1 = use apertures (not currently recommended)
use_pupil_mask	0 or 1 (SPC modes only)	1	Use SPC pupil mask (SPC modes only)
use_fpm	0 or 1	1	1 = use FPM
use_lyot_stop	0 or 1	1	1 = apply Lyot stop
use_dm1 use_dm2	0 or 1	0	Use DM pistons provided in dm1_v, dm2_v
use_field_stop	0 or 1 (HLC only)	1	1 = apply field stop

Aberrations

Parameter	Valid value or type	Default	Description
polaxis (*)	-2 = -45°in,+Yout -1 = -45°in,+Xout 0 = none +1 = +45°in,+Xout +2 = +45°in,+Yout 5 = mean of ±45°in,+Xout 6 = mean of ±45°in,+Yout 10 = mean of all axes	0	Defines axis of polarization aberrations. WFC should solve for 5, 6, or 10, then the solution evaluated separately using -1 & 1 for 5, -2 & 2 for 6, and -2, -1, 1, & 2 for 10.
use_errors	0 or 1	1	1 = use optic fabrication & alignment errors
zindex	integer ≥ 1	0	Vector of Zernike polynomial indices (Noll)
zval_m	meters RMS WFE	0	Vector of Zernike coefficients (Noll)

DM parameters

Parameter	Valid value or type	Default	Description
use_dm1 use_dm2	0 or 1	0	Use DM pistons provided in dm1_m, dm2_m
dm1_v dm2_v	volts	0	48x48 arrays containing actuator pistons
dm_sampling_m	meters	0.9906e-3	DM actuator spacing
dm1_xc_act, dm1_yc_act dm2_xc_act, dm2_yc_act	actuators	23.5	Center of wavefront on DM in units of actuators (0,0 = center of 1st actuator)
dm1_xtilt_deg, dm2_xtilt_deg dm1_ytilt_deg, dm2_ytilt_deg dm1_ztilt_deg, dm2_ztilt_deg	degrees	9.65 0 0	Rotation of DM about specified axis

Intermediate field input/output

Parameter	Valid value or type	Default	Description
end_at_fpm_exit_pupil (*)	0 or 1	0	End propagation at FPM exit pupil
end_at_fsm (*)	0 or 1	0	End propagation at FSM
end_at_exit_pupil (*)	0 or 1	0	End propagation at final image exit pupil

Focal plane mask parameters

Parameter	Valid value or type	Default	Description
use_fpm	0 or 1	1	1 = use FPM
fpm_x_offset fpm_y_offset	λ_0/D	0	Shift of FPM in λ_0/D
fpm_x_offset_m fpm_y_offset_m	meters	0	Shift of FPM in meters
fpm_z_shift_m	meters	0	Shift of FPM along optical axis
pinhole_diam_m	diameter in meters	0	Pinhole at FPM plane (if non-zero)

CVS-only parameters

Parameter	Valid value or type	Default	Description
cvs_source_z_offset_m source_y_offset_mas	meters	0	Offset the CVS source along the optical axis; positive value increases the distance to CGI
cvs_stop_x_shift_m, cvs_stop_y_shift_m	meters	0	Offset of the OTA pupil mask relative to the optical axis
cvs_stop_rotation_deg	degrees clockwise	0	Rotation of the CVS OTA pupil mask around the optical axis
cvs_jitter_mirror_x_offset, cvs_jitter_mirror_y_offset	λ_0/D	0	Offset of source caused by tilt introduced at the CVS jitter mirror
cvs_jitter_mirror_x_offset_mas, cvs_jitter_mirror_y_offset_mas	milliarcsec	0	Offset of source caused by tilt introduced at the CVS jitter mirror

Source offsets (wavefront tilt at primary, FSM tilt)

Parameter	Valid value or type	Default	Description
source_x_offset_mas source_y_offset_mas	milliarcsec	0	Offset of source caused by tilt introduced at primary mirror
source_x_offset source_y_offset	λ_0/D	0	Offset of source caused by tilt introduced at primary mirror
fsm_x_offset_mas fsm_y_offset_mas	milliarcsec	0	Offset of source caused by tilt introduced at FSM
fsm_x_offset fsm_y_offset	λ_0/D	0	Offset of source caused by tilt introduced at FSM
image_x_offset_m image_y_offset_m	meters	0	Offset of the image at the detector plane

Phase retrieval parameters (defocus & pupil imaging lenses, pinhole)

Parameter	Valid value or type	Default	Description
pinhole_diam_m	diameter in meters	0	Pinhole at FPM plane (if non-zero)
end_at_fpm_exit_pupil (*)	0 or 1	0	End propagation at FPM exit pupil
use_pupil_lens	0 or 1	0	1 = Use pupil imaging lens
use_defocus_lens	0, 1, 2, 3, 4	0	0 = no defocus lens, 1 = +28.3 waves P-V, 2 = +15.4 waves, 3 = 9.2 waves, 4 = -2.9 waves (@ 575 nm, chromatic)

User-provided masks (overrides default masks; see following section for details)

Parameter	Valid value or type	Default	Description
pupil_array	2D array	null	Overrides entrance pupil pattern at primary
pupil_mask_array	2D array	null	Overrides default SPC mask pattern (can also be used with HLC)
lyot_stop_array	2D array	null	Overrides default Lyot stop pattern
fpm_array	2D array	null	Overrides default FPM pattern
fpm_array_sampling	float	0	Sampling in meters of fpm_array
field_stop_array	2D array	null	Overrides field stop pattern (can be used with HLC or SPC)
field_stop_array_sampling	float	0	Sampling in meters of field_stop_array

Component offsets (CGI box, SPC mask, FCM, FPM, Lyot stop)

Parameter	Valid value or type	Default	Description
cgi_x_shift_pupdiam cgi_y_shift_pupdiam	-1 to +1 as fraction of pupil diameter	0	Bulk shear of CGI at FSM relative to IC
cgi_x_shift_m cgi_y_shift_m	meters	0	Bulk shear of CGI at FSM relative to IC in meters
excam_despace_m	meters	0	Increase in the spacing between the final optic and the detector
fcm_z_shift_m	meters	0	Piston of focus corrector mirror (note: only alters focus, not other aberrations)
field_stop_x_offset field_stop_y_offset	λ_0/D	0	Shift of HLC field stop in λ_0/D
field_stop_x_offset_m field_stop_y_offset_m	meters	0	Shift of HLC field stop in meters
fpm_x_offset fpm_y_offset	λ_0/D	0	Shift of FPM in λ_0/D
fpm_x_offset_m fpm_y_offset_m	meters	0	Shift of FPM in meters
fpm_z_shift_m	meters	0	Shift of FPM along optical axis
lyot_x_shift_pupdiam lyot_y_shift_pupdiam	-1 to +1 as fraction of pupil diameter	0	Lyot stop shift relative to pupil diameter
lyot_x_shift_m lyot_y_shift_m	meters	0	Lyot stop shift in meters
lyot_rotation_deg	degrees	0	Rotation of the Lyot stop about the optical axis
mask_x_shift_pupdiam mask_y_shift_pupdiam	-1 to +1 as fraction of pupil diameter (SPC only)	0	SPC pupil mask shift relative to pupil diameter
mask_x_shift_m mask_y_shift_m	meters (SPC only)	0	SPC pupil mask shift in meters
mask_rotation_deg	degrees clockwise	0	Rotation of the SPC mask about the optical axis
sm_despace_m	meters	0	Change in the default separation between the primary and secondary mirrors
spam_x_shift_m spam_y_shift_m	meters	0	Shear of the wavefront at SPAM, as a fraction of the pupil diameter; applied after mask (SPC mode only, ignored otherwise)
spam_x_shift_pupdiam spam_y_shift_pupdiam	-1 to +1 as fraction of pupil diameter	0	Shear of wavefront at SPAM, as a fraction of the pupil diameter; applied after mask (SPC mode only, ignored otherwise)

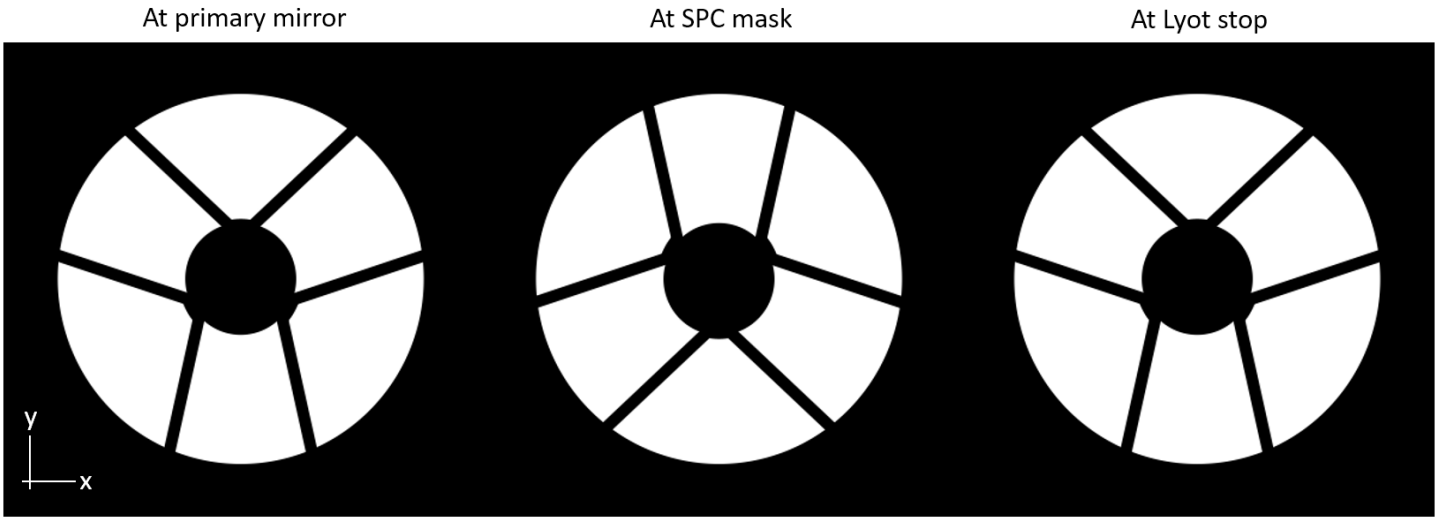
User-defined masks

This super-secret JPL version of the model now allows users to insert their own masks, passed as optional parameters. These will override the default ones. These parameters are listed in the tables above. These are provided as square 2D arrays (Numpy arrays in Python). Provided arrays are trimmed or zero-padded to match the wavefront grid size. In all cases, the wavefront in a N by N array is centered at pixel N//2 (python), fix(N/2) (IDL), and fix(N/2)+1 (Matlab).

Pupil masks (entrance pupil, SPC pupil mask, Lyot stop) do not change scale with wavelength and are defined using `pupil_array`, `pupil_mask_array`, and `lyot_stop_array`. The beam size is always 1000 pixels across in SPC modes (these are values in the Y direction for asymmetric pupils). Note that if you provide a pupil that is larger or smaller than the predefined one, that **does not** change the scale (meters/pixel) at non-pupil planes. Entrance pupils and SPC masks should be defined in their separate planes.

The focal plane masks are defined by `fpm_array` and `field_stop_array`. The sampling of focal plane masks (using `fpm_array_sampling_m` and `field_stop_array_sampling_m`) must be provided so that the scale change with wavelength is accounted for (the model will adjust the sampling of the wavefront during propagation to/from the focal plane mask or stop). You should provide these masks with samplings equal to or finer than the default sampling at the center wavelength of the band (as derived from the "Focal plane dimensions" table below). Note that typically the field stop is not used in SPC mode, but when `field_stop_array` is defined, then it is (the value of `use_field_stop` is ignored in this case). Also, there is no need to pad the array, except perhaps by a pixel, since MFTs are used.

See the images below for the orientations of pupils and focal planes.

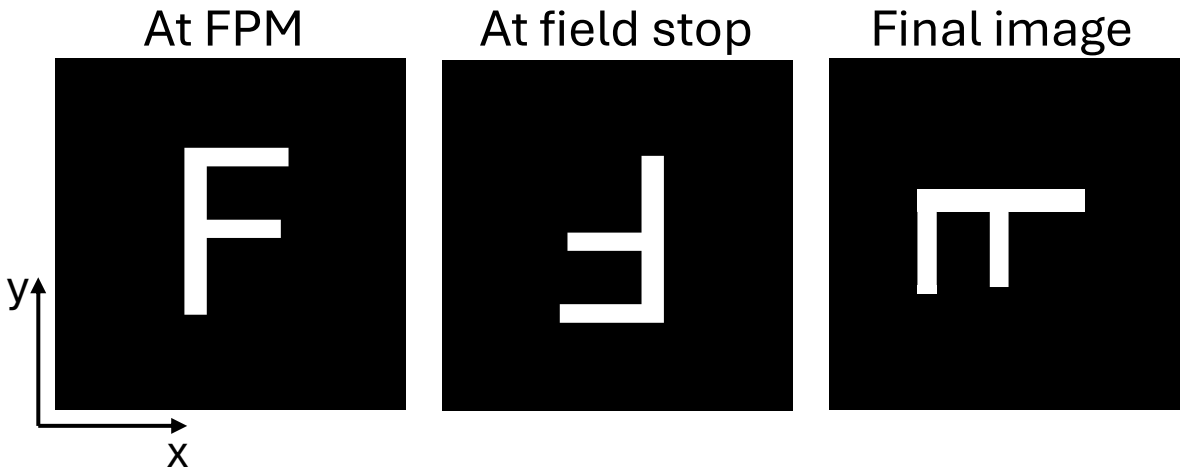


Orientation of the pupil at different pupil planes, along with the X-Y coordinate convention.

Pupil plane dimensions

Plane	SPC beam diam (pix) / grid diam (pix)	Beam diam (m)
Primary mirror	1000 pix / 2048 pix	2.363114 m
SPC mask	1000 pix / 2048 pix	0.0170 m
Lyot stop	1000 pix / 2048 pix	0.0170 m

NOTE: When a defocus lens (not pupil imaging lens) is used, then the grid diameters are 2x larger than listed here (the beam diameters remain unchanged).



Orientation at the image planes, along with the local X-Y coordinate convention.

Focal plane dimensions

Plane	SPC Band 2 sampling at λ = 660 nm	SPC Band 3 sampling at λ = 730 nm	SPC sampling λ/D	SPC grid diam (pix)
FPM	1.03918e-5 m	1.14940e-5 m	1000 / 2048 = 0.488281	2048
Field stop	1.34273e-5 m	1.48513e-5 m	1000 / 2048 = 0.488281	2048

*NOTE: When a defocus lens (not pupil imaging lens) is used, then the grid diameters **at the field stop** are 2x larger and the samplings are 2x smaller than listed here.*

Change log:

V0.9 – 21 June 2020: Initial release of the Phase C prescription

V0.9.1 – 29 June 20202: rotated SPC pupils & Lyot stops to be consistent with HLC

V0.9.2 – 7 July 2020: updated the HLC occulter files

V1.0 – 22 Sept 2020: updated HLC FPM again; final release.

V1.0.1 – 5 Oct 2020: added SPC Band 2 flat-wavefront and dark hole DM solutions to *examples* directory

V1.0.2 – 28 Oct 2020: added optional parameters to pass masks (for AJ); switched to 2048x2048 grid sizes for SPC, except when running phase retrieval images

V1.2 – 12 April 2021: increased distance from TT Fold to FSM to move the pupil 6 mm from the SPC mask (and Lyot stop) to represent telescope pupil defocus caused by the TCA; increased the pupil imaging lens doublet separation by 0.2 mm to produce a good pupil image on the detector (will be implemented in real-life); added code to subtract quadratic phase term in final image due to conjugate pupil not being at the front focus of imaging lens (only applied when NOT using PIL or defocus lenses); switched to MFT-based propagation to/from HLC FPM at high sampling to avoid interpolation errors; added spc-mswc mode (provided by A.J. Riggs) for multi-star wavefront control modeling; updated error maps; removed user-specified lam0 and lambda0_m parameters (use only hard-coded values); changed default wavefront grid sizes, including getting rid of intermediate grid size changes; added secondary mirror despace parameter; read list of available HLC FPM files from table; removed propagation to fold 4 (not significant); added TO_PLANE option to thick lenses; added options for using Band 1 with the spc-wide and spc-mswc; removed the *input_field_rootname* and *output_field_rootname* options, since the compact model is no longer fully compatible with the full model (because it lacks the pupil defocus); removed the *polaxis* option from the compact model; HLC designs for bands 2-4 were added as well as a rotated SPC-Spec design (from AJ Riggs)

V1.2.1 – 14 May 2021: added option to use field_stop_array with HLC

V1.2.2 – 20 May 2021: increased grid size for HLC to 2Kx2K when using phase retrieval lenses; updated aperture sizes

V1.2.3 – 15 July 2021: added dm_sampling_m as optional parameter

V1.2.4 – 3 Aug 2021: added HLC FPM files for filter 3g; updated manual to discuss HLC designs for Bands 2, 3, & 4.

V1.2.5 – 27 Sept 2021: added use_lens_errors parameter

V1.2.6 – 7 Oct 2021: (change only to Python version) added save_ref_radius optional parameter (Python only, used by CGISim for phase retrieval studies).

V1.2.7 – 21 Oct 2021: if SPC is used and use_pupil_mask is zero, then the fold mirror aberrations will be used instead of the SPC substrate's; fixed bug in Matlab version that used the wrong name for num2str when reporting a wavelength mismatch for the HLC FPM.

V1.2.8 – Added option for spc-spec_band2_rotated; fixed mask orientations for rotated SPC

V1.2.9 – Added options to shift the beam at the SPAM and shift the image on the detector; allow user-provided SPAM masks whether using SPC or HLC

V1.2.9a – Fixed FPM orientations for HLC bands 2-4; changed PIL separation to 0.5 mm

V1.2.9b – Changed spam_*shift_* parameters to work only in SPC mode, otherwise they are ignored; fixed bug in Matlab version for reading PIL error maps

V1.3 – 14 June 2022 - Added option (use_cvs) to select CVS instead of Roman+TCA as front end; measured CVS OAP aberrations included; added options for CVS source Z offset and CVS OTA pupil mask X,Y,Z offsets and rotation about optical axis; added option for CVS jitter mirror tilt; added option to rotate SPC pupil masks and Lyot stop; pupil and focal plane masks now account for sampling changes caused by motion of secondary, CVS source, or FCM, which changes magnification; updated parameters for pupil imaging and defocus lenses with measured values (provided by David Marx).

V1.4 – 15 Oct 2022 – Added ZWFS; fixed orientation of rotated SPC; fixed orientation of CVS stop

V1.5 – 7 March 2023 – Phase retrieval parameters updated as provided by David Marx: updated error files for defocus and PIL lenses to remove defocus term; updated lens curvatures; added excam_despace_m parameter

V2.0 – August 2024 – Incorporated a new DM model that includes measured flight device characteristics, including actuator gains, coupling, dead actuators, influence functions (appropriate for each DM), and DM surface errors; DM strokes are now specified in volts rather than meters of piston; the image output orientation now matches that of real CGI images, which involves a transpose and rotation relative to prior versions of the model; the FOCM (focus corrector mirror) is now called FCM, so it is now controlled using fcm_z_shift_m; more baffles have been added, include a new one in front of the color filter mechanism that was installed after thermal vac testing (these are off by default, and are activated by setting use_aperture = 1; the use_lens_errors option has been removed; prior versions had an artificial amount of low order wavefront aberrations added to provide a total wavefront error of 76 nm RMS at the entrance to CGI in addition to the measured surface errors, the formal system requirement – that has been removed as recent analyses suggest that the error will be ~40 nm RMS.