Nancy Grace Roman Space Telescope
Technical Report Roman-STScI-000433

`WFIsim`: The Roman Telescope Branch Wide-Field-Instrument Simulator

A. Bellini, T. D. Desjardins, S. Casertano, & R. E. Ryan Jr.

November 11, 2022

Abstract

We describe a software package, `WFIsim`, that has been developed by the Roman Telescope Branch to simulate realistic Nancy Grace Roman Space Telescope Wide-Field Instrument data, with the primary goal of scientifically validating the Roman calibration pipeline. `WFIsim` is written in Fortran, and a python wrapper will interface with the Calibration Reference Data System and the calibration pipeline.

In a typical run, the user specifies a set of parameters in a configuration file and provides input catalogs of sources, effective Point-Spread-Function models and calibration files. `WFIsim` then goes through a series of steps to simulate the astronomical scene, the level-1 raw exposures and level-2 calibrated exposures. It can also output diagnostic information in the form of FITS or ASCII files.

`WFIsim` is designed to be modular and flexible, and new capabilities and functionalities will be added as we know more about the WFI and the Roman mission. The program can be useful for detector-characterization studies and calibration purposes. This report describes how `WFIsim` operates, all the input files it needs, and guides the user through a full simulated run.

1 Introduction

`WFIsim` is a `Fortran` program that simulates Nancy Grace Roman Space Telescope's (Roman) Wide-Field-Instrument (WFI) exposures in imaging mode. The program was originally developed for a Roman astrometric investigation aimed at quantifying the achievable precision of the geometric-distortion solution of the WFI using *Gaia* stars as a reference (see Bellini 2018). The capabilities of the software have since been considerably extended to simulate realistic WFI imaging observations that will be primarily used to scientifically validate the Roman calibration pipeline.

Copyright ©2022 The Association of Universities for Research in Astronomy, Inc. All Rights Reserved.

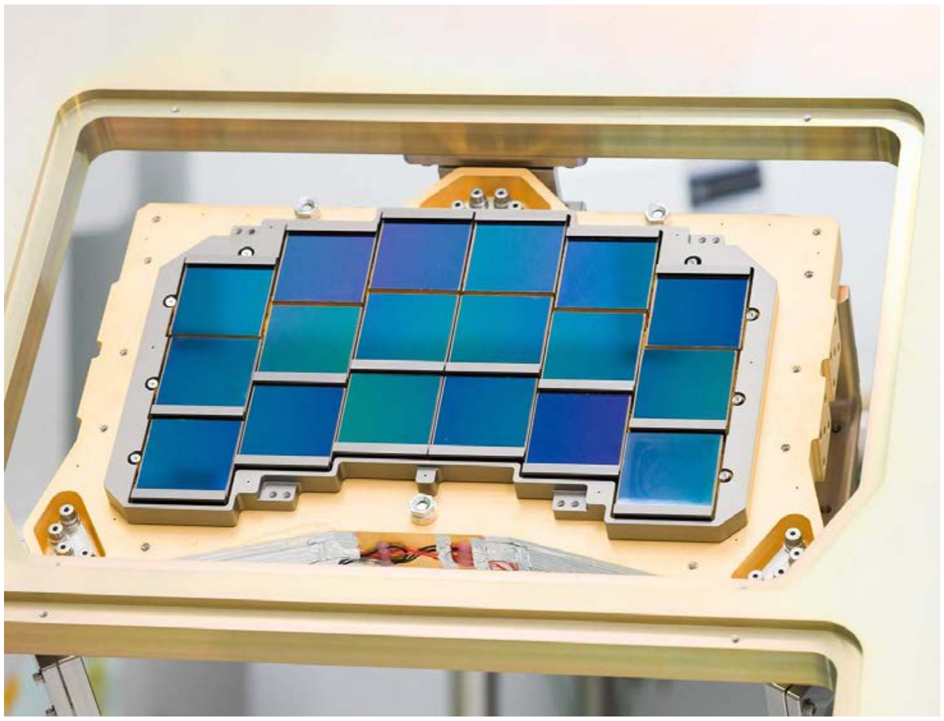


Figure 1: The Engineering Test Unit of the Roman WFI focal-plane mount with the 18 H4RG-10 SCAs arranged in a slightly curved 6×3 array. Figure taken from `https://roman.gsfc.nasa.gov/science/WFI_technical.html`.

The Roman WFI consists of 18 sensor-chip assemblies (SCAs) manufactured by Teledyne[1], which act as the main light-sensitive device used for scientific and guiding purposes. Each SCA comprises of 4096×4096 resolution elements (pixels), with a 4-pixel-wide, unilluminated boundary on each side, often called the reference pixels, used for calibration purposes. The focal-plane array (FPA) is arranged in a slightly curved 6×3 grid of SCAs (see Fig. 1). With a pixel scale of about 110 mas pixel^{-1}, the WFI has a total light-sensing area of $18 \times 4088 \times 4088 \approx 300$ megapixels, or 0.281 square degrees.

The code is mainly written in `Fortran 90`, but it also contains legacy subroutines (especially for FITS input/output) in `FORTRAN77` and it makes use of a few `Fortran 2003` modules. To maintain full backwards compatibility, the source code closely follows the `FORTRAN77` fixed format. The reason using `Fortran` is twofold. First, `WFIsim` takes advantage of state-of-the-art simulation and data-reduction routines written in `Fortran` for *Hubble Space Telescope* (*HST*) data. Additionally, the sheer size of the WFI detector makes simulating full-frame images computationally demanding, and a high-level programming language such as `Fortran`, which is particularly suited for scientific and numerical computation, makes it so that a full-frame simulation can be run within a reasonable amount of time.

Roman data will be exclusively available through Advanced Scientific Data Format

[1]`https://www.teledyne-si.com/products-and-services/imaging-sensors/hawaii-4rg`

(ASDF) files (Greenfield, Droettboom & Bray 2015). Since at present there are no `C` or `Fortran` ASDF input/output routines, `WFIsim` makes use of Flexible Image Transport System (FITS; Wells, Greisen & Harten 1981) files for input and output. A python wrapper of `WFIsim` will interface with the Calibration Reference Data System (CRDS) server to retrieve calibration ASDF files, convert them into FITS format for `WFIsim`, and convert `WFIsim`'s output FITS files into ASDF format.

`WFIsim` only simulates WFI images at the detector level: the user is in charge of providing a list of input sources (stars and/or galaxies) that will be added to the simulated images, as well as calibration reference files (in FITS format for the `Fortran` code).

The source code is fully contained into a single file (WFIsim.F) that can be compiled with gfortran using the following shell command:

```
> gfortran WFIsim.F -fopenmp -O3 -o WFIsim.e
```

where the argument "-fopenmp" informs the compiler to use the OpenMP application programming interface specification for parallel programming[2], while "-O3" activates aggressive compiler optimization[3] to improve performance. The ".F" extension of the source code activates the pre-processor steps during compilation. The program has been tested with the following gfortran versions: 4.9, 5.5, 6.4, 7.3, 7.4, 8.1, 8.2 for Linux, and 7.2, 11.1 for macOS. The "–version" and "–help" are the only command-line arguments currently available.

The structure of this technical report is as follows. Section 2 focuses on the input files and their format, and describes the available parameters. A description of the program itself and of its main subroutines is detailed in Sect. 3, together with a step-by-step example run of `WFIsim`. Performance, caveats and plans for future improvements are listed in Sect. 4. Finally, Sect. 5 summarizes our conclusions.

2 The input files

Among the several possible input files, the only mandatory one is the configuration file (hereafter `IN.WFIsim`, but note that the filename is arbitrary), which is passed to `WFIsim` as command-line argument. All the other input files are optional and are defined within `IN.WFIsim`.

2.1 The configuration file

`IN.WFIsim` is an ASCII file containing a list of parameters and associated values, one parameter per line. When a parameter is omitted, `WFIsim` assumes the default value. In the current version of `WFIsim` (0.7.0-alpha), the user can set up to 63 parameters. The example

[2]`https://www.openmp.org`.
[3]`https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`.

configuration file bundled with `WFIsim` lists all available parameters and contains a brief description of them as a quick reference guide.

At execution time, `WFIsim` parses each line of the input configuration file and looks for *exact* keyword matches. `WFIsim` is very strict with respect to the syntax used. For instance, valid entries are, e.g.: "`SAVE_CRMASK`=Y", "`SATVAL`=80000.0", "`POINTING_MODE`=RD" while invalid entries are, e.g.: "`SAVE_CRMASK`= Y" (note the space after "="), "`SATVAL`=qwerty", "`POINTING_MODE`= ".

In the following, we provide a full description of the 63 parameters. For clarity, the list is broken down into groups based on different topics.

- Global parameters.

  `MODE` specifies one of the three main ways `WFIsim` can simulate WFI data. Valid options are: "LL", "L1", and "L2". The value "LL" is used when the user only needs to create and output the astronomical scene, in units of $e^{-}\,s^{-1}$, without any detector signature. By default, the output astronomical scene is noiseless, but Poisson noise can be optionally added to it using the `ADD_POISSON_NOISE_TO_LL` parameter (see below). The value "L1" is used to produce the raw level-1 images (hereafter, L1). L1 images are 16-bit integer data cubes in analog-to-digital units (ADUs), containing the uncalibrated resultant frames of an exposure (a "ramp"). The value "L2" is used to calibrate the simulated L1 data into level-2 images (hereafter, L2). L2 images are 32-bit float arrays in units of $e^{-}\,s^{-1}$, and have detector-level calibrations (e.g., dark current) applied. L2 images are rate images that are the result of the ramp-fitting process. (Example: `MODE`=L1; the default value is LL.)

  `VERBOSE` is the amount of information that is displayed on screen. Current accepted values are 0, 1 and 2, from minimum to maximum verbosity. (Example: `VERBOSE`=2; the default value is 0.)

- Image parameters.

  `SCA_NUMBER` defines how many SCAs are to be simulated, and is expected to be an integer value from 1 to 18. In its current implementation, `WFIsim` will simulate all the SCAs ranging from 1 to `SCA_NUMBER`. (Example: `SCA_NUMBER`=18; the default value is 1.)

  `LOFLAG` is the value in $e^{-}\,s^{-1}$ assigned to bad pixels or to pixels that saturate in the very first resultant in L2 images. It is used to indicate pixels with various problems. (Example: `LOFLAG`=−70.0, the default value is −99.0.)

- Pointing parameters.

`POINTING_MODE` specifies how user-provided input positions should be treated. It can be of two types: “XY” when the coordinates are in units of pixels on a tangent-point projection, rectified Cartesian plane, and “RD” when the coordinates are in degrees on the celestial sphere, along R.A. and Dec. `POINTING_MODE` also affects the units of other parameters like, e.g., dither offsets (see below). (Example: `POINTING_MODE`=RD; the default value is XY.)

`MXREF` is only used when `POINTING_MODE`=XY, and defines the X pixel value on the reference frame of the input source catalogs where the nominal WFI aperture (i.e., the center of the focal-plane coordinate system, see., e.g., the red cross in Fig. 7) is placed. (Example: `MXREF`=20000; the default value is 0.)

`MYREF` is similar to `MXREF` but for the Y pixel location of the aperture on the reference frame. (Example: `MYREF`=18000; the default value is 0.)

`MRREF` is similar to `MXREF`, but it is only used when `POINTING_MODE`=RD, and it defines the R.A. coordinate of the aperture on the celestial sphere. Its units are degrees along R.A. (Example: `MRREF`=276.45); the default value is 0.

`MDREF` is similar to `MRREF` but for the Dec. location of the aperture. (Example: `MDREF`=-82.72; the default value is 0.)

`XDITHER` allows for dither maneuvers along the horizontal axis (X or R.A., according to how `POINTING_MODE` is set) relative to the location of the nominal aperture of the WFI (`MXREF` or `MRREF`). It is similar to the *HST*'s `POSTARG` command (POSition TARGet). Units are pixels when `POINTING_MODE`=XY, and arcsec when `POINTING_MODE`=RD. Note that X and R.A. increase towards opposite directions. (Example: `XDITHER`=20.5; the default is 0.)

`YDITHER` is similar to `XDITHER` but for a vertical-axis dither with respect to either `MYREF` or `MDREF`. (Example: `YDITHER`=0.3; the default is 0.)

`ROLL_ANGLE` defines the pointing orientation around the aperture, in degrees, counterclockwise with respect to the +Y or North direction (it depends on to how `POINTING_MODE` is set). Note that the `ROLL_ANGLE` orientation is *not* the spacecraft orientation, which is instead defined around the telescope optical axis. As a reference, a `ROLL_ANGLE` of zero degrees indicates that the spacecraft has rolled about the +V1 axis such that the angle from the Celestial North direction and the +V3 axis is 60 degrees. (Example: `ROLL_ANGLE`=35.8; the default value is 0.)

`ADD_DITHER_NOISE` is used to simulate uncertainties in the exact placement of the nominal aperture on the reference system of the input source catalogs. Valid entries are “Y” for yes and “N” for no. (Example: `ADD_DITHER_NOISE`=N, which is the default value.)

`DITHER_NOISE_VALUE` sets the uncertainty on the placement of the aperture in the coordinate system defined by the input source catalogs. Units are either pixels or arcsec, depending on how `POINTING_MODE` is set. The noise itself is Gaussian with sigma equal to the parameter value. (Example: `DITHER_NOISE_VALUE`=0.04; the default value is 0.)

- Astronomical-scene parameters.

`ADD_SCENE` allows the user to specify the three astronomical sources of signal in the simulation: (i) stars, (ii) galaxies, and (iii) sky background. Valid entries are "Y" for yes and "N" for no. When `ADD_SCENE`=N (the default value), `WFIsim` can be used to simulate dark exposures. (Example: `ADD_SCENE`=Y; the default value is N.)

`PSFPATH` is the full path of the directory containing the effective point-spread-function (ePSF) models. The ePSF directory is expected to contain at least one subdirectory with a valid filter name, e.g., "F062" (see below under "`FILTER`"). The filter subdirectory must contain ePSF models with specific names, sizes and format. More details in Sect. 2.2. (Example: `PSFPATH`=/user/username/roman/epsfs/; the default value is NULL.)

`FILTER` defines which filter-dependent ePSF models should be used for the simulation. Valid entries are: "F062", "F087", "F106", "F129", "F146", "F158", "F184", and "F213". (Example: `FILTER`=F146, which is the default value.)

`PSFRAD_FACTOR` is a scaling factor used to calculate how far from the center of a source `WFIsim` needs to go to add its flux to the astronomical scene. `PSFRAD_FACTOR` is used to improve performance. Typical values for `PSFRAD_FACTOR` are in the range 1.0–10.0; the higher the number the farther from its center the flux of a source is added. More specifically, flux is added until $p(r) < K$, where $p(r)$ is the total-flux-scaled PSF at distance r from a star's center, and $K = \mathrm{PSFRF} \times \mathrm{FST} \times \sigma_{BK}/100$, where PSFRF=`PSFRAD_FACTOR`, FST=`FAINT_SIGMA_THRESHOLD`, and σ_{BK} is the square root of the total sky background value. (Example: `PSFRAD_FACTOR`=3.5; the default value is 5.0.)

`ADD_STARS` allows the user to specify an input star catalog. Accepted values are "Y" for yes and "N" for no. (Example: `ADD_STARS`=Y; the default value is N.)

`STARCAT_NAME` defines the full path of the input star catalog, and it is only used when `ADD_STARS`=Y (example: `STARCAT_NAME`=/user/username/stars.dat; the default value is NULL). The catalog itself must contain one line per source. Each source must be characterized by three values (space or tab separated): the first two are the X and Y pixel positions (when `POINTING_MODE`=XY) or R.A. and Dec.

Table 1
Example of Input Star Catalog

X coordinate	Y coordinate	Instr. mag.
21000.7862	721.3187	−8.1882
11098.6884	6740.2022	−6.7828
487.4996	10784.7033	−9.5136
4737.9131	14254.1178	−3.4761
37073.6156	9145.4061	−10.1521
4732.1956	20477.7019	−9.3820
46186.2382	32847.2495	−7.3234
8639.3426	33908.6764	−12.1250
42532.7870	10734.8219	−7.3248
45962.7407	14948.9363	−8.3152
...	...	...

degree positions (when `POINTING_MODE`=RD), while the last value is the instrumental magnitude[4]. If the catalog file contains header and/or commented lines, they must start with "#". Table 1 shows an example of an input star catalog.

`NSTARS` is used to limit the number of sources in the input star catalog that will be simulated to the first `NSTARS` in the list, in case the user needs to simulate fewer stars than those present in the catalog. `NSTARS` can be set to the value of −1 to inform `WFIsim` that the entire input star catalog should be used. (Example: `NSTARS`=500 000; the default value is 0.)

`ADD_GALAXIES` is similar to `ADD_STARS` but for galaxies. (Example: `ADD_GALAXIES`=N, which is the default value.)

`GALCAT_NAME` is similar to `STARCAT_NAME` but for the full path of the input galaxy catalog (example: `GALCAT_NAME`=/user/username/galaxies.dat; the default value is NULL). `WFIsim` simulates galaxies using simple Sérsic profiles, and the input galaxy catalog must contain one line per galaxy with seven space- or tab-separated values per line. The units of many of these values follow how `POINTING_MODE` is set, and are either pixels or degrees/arcsecs. The column-by-column information of the input galaxy catalog is as follows: (1) the X (pixel) or R.A. (degree) position of the source; (2) the Y (pixel) or Dec. (degree) position of the source; (3) the effective radius $R_{\rm e}$ (in pixel or arcsec), i.e., the radius containing half of the galaxy flux; (4) the Sérsic parameter n; (5) the ellipticity, defined as $(a-b)/a$, where a and b are the major and minor axes of the galaxy, respectively; (6) the inclination angle, in degrees, counterclockwise from the +Y or North direction; and (7) the instrumental magnitude within the effective radius. As for the input star catalog,

[4]Instrumental magnitudes are defined as $-2.5 \times \log(\mathrm{Flux})$, where Flux is the total source flux in $e^- s^{-1}$.

header or commented lines must start with "#". An example of an input galaxy catalog is given in Table 2.

`NGALS` is similar to `NSTARS` but for the galaxy catalog. (Example: `NGALS`=250000; the default value is 0.)

`FAINT_SIGMA_THRESHOLD` sets the minimum number of sigmas above the sky background level that the central pixel of a star must have to be added to the astronomical scene. The parameter is internally divided by 50 when considering galaxies. While the galaxy-related additional divider value cannot be set in `IN.WFIsim`, the user can still change it by modifying the value of the variable `_faint_sigma_gals_` in the preamble of the source code and recompiling it. Similarly to `PSFRAD_FACTOR`, `FAINT_SIGMA_THRESHOLD` is also used to increase performance. (Example: `FAINT_SIGMA_THRESHOLD`=1.5, which is the default.)

`SAVE_INPUT_POSITIONS` allows the output of space-separated ASCII files (one per SCA) listing SCA pixel raw positions, meta-frame positions, and magnitudes (for stars) or Sèrsic parameters (for galaxies) of all sources actually added to the astronomical scene. The meta frame is the tangent-plane projection of the astronomical scene around the nominal aperture, shifted to (X, Y)=(13 315, 10 225) so that sources always map into positive pixels (see Sect. 3.1.1 for details). The astronomical scene on the meta frame can be saved as a FITS file, see `SAVE_META`. Valid options are "Y" for yes and "N" for no. `WFIsim` outputs distinct catalogs for stars (extension `_str.poslog`) and galaxies (extension `_gal.poslog`). (Example: `SAVE_INPUT_POSITIONS`=Y; the default value is N.)

`ADD_BACKGROUND` is used to add a sky background to the astronomical scene. The sky background can be either flat, i.e., all the pixels of the scene have the same sky-background value, or the user can provide input FITS files (one per SCA) with custom sky background. Valid entries are "Y" for a flat sky background, "N" for no sky background, and "I" for user-provided 32-bit float 4088×4088-pixel FITS files. For the last option, the file names of the sky-background files must be: BKGRD_SCAXX.fits, where XX is the SCA number. (Example: `ADD_BACKGROUND`=I; the default value is N.)

`BACKGROUND` is used to define the sky background in terms of a flat value in $\mathrm{e^- \, s^{-1}}$. It is only used when `ADD_BACKGROUND`=Y. (Example: `BACKGROUND`=0.9; the default value is 0.)

`BKGRDPATH` indicates the full path of sky-background FITS images. Is only used when `ADD_BACKGROUND`=I. (Example: `BKGRDPATH`=/user/username/background/; the default value is NULL.)

`ADD_JITTER` allows the user to add jitter effects to the simulation. The jitter kernel is calculated as a two-dimensional Gaussian characterized by three quantities:

Table 2
Example of Input Galaxy Catalog

R.A. coordinate	Dec. coordinate	R_e	n	Ellip.	Inclination	Instr. mag.
79.946693993884	30.304403429590	1.0066	7.8211	0.4070	196.1466	3.2460
80.495081332972	30.003675208406	0.6341	0.6913	0.3460	120.0572	6.2830
79.430667503674	30.244260779260	0.9850	9.1759	0.0993	313.5301	3.6575
79.743439120820	30.029706240067	0.5575	9.2185	0.5476	88.0054	5.9398
79.911503168124	29.560539202396	1.5316	4.3906	0.5088	120.1617	1.3893
79.742728330427	30.410821500130	0.7333	0.7600	0.6049	79.1200	5.9664
79.269550119311	29.666730007332	0.9275	7.8184	0.5676	186.2957	4.3363
80.956282565609	29.701821419014	0.9866	6.7400	0.5531	174.0041	3.4657
80.347163626545	29.561815180384	0.4825	0.5256	0.4890	41.5214	7.0135
79.163556937585	29.458213376647	0.2200	2.8474	0.0360	3.4416	8.7038
...	...	...	...	...	...	...

semi-major axis, semi-minor axis and orientation. The values of the semi axes are randomly drawn from a Gaussian distribution with a sigma of 14 mas, which is a somewhat conservative value for the expected jitter RMS of Roman. The current version of `WFIsim` does not allow the user to change the jitter RMS in the `IN.WFIsim` file, but this can still be done in the preamble of the source code by changing the `_jitter_rms_` parameter. The orientation is randomly chosen between 0 and 180 degrees, counterclockwise from the X axis. The jitter kernel is the same for all SCAs, and is convolved to the astronomical scene. Note that the current implementation of the jitter kernel does not allow for jitters in roll but only in yaw and pitch. Valid entries are "Y" for yes and "N" for no. (Example: `ADD_JITTER`=Y; the default value is N.)

`SAVE_JITTER_KERNEL` allows to output a $\times 10$-supersampled representation of the central 3×3 pixels of the jitter kernel to a FITS file. Valid entries are "Y" for yes and "N" for no. (Example: `SAVE_JITTER_KERNEL`=N, which is the default.)

- Level 1 specification parameters.

`MATABLE` is the multi-accum table that defines how an observation is broken down into reads and resultant frames (on-board averages of a set of reads). In `WFIsim` the multi-accum table is represented as a string of characters of maximum length 510, one character per read. Allowed characters are: "I", "R", "S", and "T".

I. When used, this character must be the first of the `MATABLE` string. It tells `WFIsim` to treat the first read as the reference read. The reference read is then subtracted from each resultant and is used to improve on-board compression of the data.

S. It defines a skipped/dropped read.

R. It informs `WFIsim` that this read needs to be read and added to the buffer.

T. This character indicates a read that is read and added to the buffer, and is the last read of a set of reads that should be averaged together into a resultant frame. When `WFIsim` reaches a "T" character of the `MATABLE`, the resultant frame is obtained by dividing the buffer by the number of reads in the set, and subtracting the reference read when present.

As an example, let us consider the multi-accum table defined by the following line in `IN.WFIsim`: `MATABLE`=ITRTRRRTSSRRRRRT. In this case, the first read is used as a reference read, followed by 4 resultants made up by 1, 2, 4, and 6 reads, respectively. Two reads are skipped before the beginning of the last resultant. The number of reads in a set can be 1 to 13, 16, 32, and 64. (The default value of the `MATABLE` is IRRT.)

`CLOCK_SPEED` is the time needed by the on-board electronics to read a single read, in seconds. (Example: `CLOCK_SPEED`=3.04, which is the default value for imaging.)

`ADD_CR` is used to simulate the impact of cosmic rays (CRs) to each read. The rate of CR events per SCA (80 per second) is estimated according to Robberto (2010). The length of the CR traces is modeled following Miles et al. (2021). The charge per unit length is linearly randomized between 100 and 1000 electrons per μm. Rate, length and deposited charge values will be updated as soon as the study on the impact of CRs on the James Webb Space Telescope cameras will be made public. Valid entries are "Y" for yes and "N" for no. (Example: `ADD_CR`=N, which is the default value.)

`USEFLAT` allows the user to define input master-flat images (one per SCA). These are 32-bit float 4088×4088-pixel FITS files. Valid entries are "Y" for yes and "N" for no. The file names must be: FLAT_SCAXX.fits, where XX is the SCA number. (Example: `USEFLAT`=Y; the default value is N.)

`FLATPATH` is the full path containing the input master-flat images. This parameter is only used when `USEFLAT`=Y. (Example: `FLATPATH`=/user/username/flat/; the default value is NULL.)

`DARKCURRENT` is used to define a constant dark-current rate in e^{-} $s^{-1}$for all pixels when no input master-dark files are provided. (Example: `DARKCURRENT`=0.005, which is the default value.)

`USEDARK` is similar to `USEFLAT` but for user-provided, input master-dark FITS files. When set to "Y", the `DARKCURRENT` parameter is ignored. At odds with the calibration pipeline, `WFIsim` uses rate images for the darks, in units of e^{-} s^{-1}, instead of data cubes having one layer per resultant. Master-dark files are $4096 \times$

4096-pixel, 32-bit float FITS. The file names must be: DARK_SCAXX.fits, where XX is the SCA number.(Example: `USEDARK`=N, which is the default value.)

`DARKPATH` is similar to `FLATPATH` but for the path containing the input master-dark FITS files. (Example: `DARKPATH`=/user/username/dark/; the default value is NULL.)

`USEBIAS` is similar to `USEFLAT` but for user-provided, input master-bias FITS files. When `USEBIAS`=N, a hardcoded constant value of 400 electrons per pixel is added during the simulation of level-1 data. This value can be changed in the preamble of the source code using the `_pedestal_value_` parameter. Master-bias files are 4096×4096-pixel, 32-bit float FITS files, in units of electrons. The file names need to be of the form: BIAS_SCAXX.fits, where XX is the SCA number.(Example: `USEBIAS`=Y; the default is N.)

`BIASPATH` is similar to `FLATPATH` but for the path containing the input master-bias FITS files. (Example: `BIASPATH`=/user/username/bias/; the default value is NULL.)

`USEGAIN` is similar to `USEFLAT` but for the user-provided, input gain FITS files. When `USEGAIN`=N, a constant value of 2 is used for all pixels (this value can be changed in the preamble of the source code using the `_default_gain_` parameter). Gain files are 4096×4096-pixel, 32-bit float FITS. The file names must be: GAIN_SCAXX.fits, where XX is the SCA number. (Example: `USEGAIN`=Y; the default value is N.)

`GAINPATH` is similar to `FLATPATH` but for the path containing the input gain FITS files. (Example: `GAINPATH`=/user/username/gain/; the default value is NULL.)

`USECNL` is similar to `USEFLAT` but for user-provided, input *inverse* classic-non-linearity (CNL) FITS files. These files are 64-bit float, $4096 \times 4096 \times N$-pixel data cubes, where N is the order of the polynomial of the inverse CNL solution: $\sum_{i=0}^{N} a_i x^i$. The layers of the data cube must start with the a_0 coefficient. Allowed polynomial order are currently in the range 7–11. The inverse CNL FITS files must be named: LNC_SCAXX.fits, where XX is the SCA number and LNC is a mnemonic pun to indicate the inverse polynomial solution of the CNL correction. Note that only the inverse CNL files are used by `WFIsim`. (Example: `USECNL`=Y; the default value is N.)

`CNLPATH` is similar to `FLATPATH` but for the path containing the input FITS files with the inverse CNL coefficients. (Example: `CNLPATH`=/user/username/cnl/; the default value is NULL.)

`USEDQ` is similar to `USEFLAT` but for user-provided, input data-quality FITS files. Data-quality files contain flags that are used to identify pixels that are good, bad, hot, etc. Pixels can be affected by multiple flags up to a total of 31. Data-quality FITS files are made up of 32-bit unsigned integer, 4096 × 4096 pixels. Since the exact mapping between bit values and data-quality flags for Roman has not being finalized yet, the current version of `WFIsim` assumes the following: good pixels have a value of zero, unexposed pixels and 33rd-amplifier pixels (i.e., the reference pixels, more in Sect. 3.2) have a value of 2^{31}. Pixels with any other value are treated as bad. File names must be of the form: DQ_SCAXX.fits, where XX is the SCA number. (Example: `USEDQ`=N, which is the default value.)

`DQPATH` is similar to `FLATPATH` but for the path containing the input data-quality FITS files. (Example: `DQPATH`=/user/username/dq/; the default value is NULL.)

`SATVAL` defines a constant saturation level for all pixels, in units of electrons. The saturation level is assumed to be unaffected by classic non-linearity effects. If the user supplies saturation reference files through the `USESAT` and `SATPATH` parameters, the value of `SATVAL` will be ignored. (Example: `SATVAL`=65000.0, the default value is 80000.0.)

`USESAT` is similar to `USEFLAT` but for user-provided, input saturation FITS files. Saturation files are 32-bit float, 4096 × 4096-pixel FITS in units of electrons. The file names must be: SAT_SCAXX.fits, where XX is the SCA number. When `USESAT`=Y, saturation files will supersede the `SATVAL` parameter. As for the `SATVAL` case, the saturation level is assumed to be unaffected by classic non-linearity effects. (Example: `USESAT`=N, which is the default value.)

`SATPATH` is similar to `FLATPATH` but for the path containing the input saturation FITS files. (Example: `SATPATH`=/user/username/sat/; the default value is NULL.)

`CDS_NOISE` indicates the amount of correlated double sampling (CDS) noise between two consecutive reads, in units of electrons. Its value, divided by $\sqrt{2}$, is used to define a constant read noise for all pixels, since this is the only source of read noise currently used by `WFIsim`. `CDS_NOISE` is superseded by user-provided read-noise input FITS files using `USERN` and `RNPATH`. (Example: `CDS_NOISE`=15, which is the default value.)

`USERN` is similar to `USEFLAT` but for user-provided, input read-noise FITS files. Read-noise files are 4096 × 4096-pixel, 32-bit float FITS. The file names must be: RN_SCAXX.fits, where XX is the SCA number. (Example: `USERN`=N, which is the default value.)

`RNPATH` is similar to `FLATPATH` but for the path containing the input read-noise FITS files. (Example: `RNPATH`=/user/username/rn/; the default value is NULL.)

- Output parameters

`OUTPUT_ROOTNAME` sets the root name of all output files, e.g., `OUTPUT_ROOTNAME`=image; the default value is "WFIsim_out".

`SAVE_LL` turns on ("Y") or off ("N") the option to save the astronomical scene in $e^- s^{-1}$ to disk as FITS images, one per SCA, with extension `_LL_SCAXX.fits`, where XX is the SCA number. (Example: `SAVE_LL`=Y, which is the default value.)

`ADD_POISSON_NOISE_TO_LL` adds some noise to the LL images just before they are saved to FITS files (`SAVE_LL`=Y). The noise is computed as the Poisson noise the scene would produce during the full exposure as it would on a CCD. This option only affects the LL output[5] and might be useful when the user wants to run source-measuring programs requiring noisy images on `_LL` images. Valid entries are "Y" for yes and "N" for no. (Example: `ADD_POISSON_NOISE_TO_LL`=N, which is the default value.)

`SAVE_L1` is similar to `SAVE_LL` but for L1 FITS images, one per SCA, in the form of 16-bit integer data cubes of $4224 \times 4096 \times N_{\rm res}$ pixels, where $N_{\rm res}$ is the number of resultants defined in `MATABLE`. This parameter is ignored when `MODE`=LL. L1 images have extension `_L1_SCAXX.fits`, where XX is the SCA number. (Example: `SAVE_L1`=N, which is the default value.)

`SAVE_CRMASK` allows the output of the CR masks in form of FITS files. These are 16-bit integer data cubes of $4224 \times 4096 \times N_{\rm res}$ pixels, one per SCA. The values of the pixels in the data cubes represent the read number in which they are first hit by a CR (only the first hit is recorded). The file extension is `_crmask_SCAXX.fits`, where XX is the SCA number. Valid entries are "Y" for yes and "N" for no. This parameter is ignored when `MODE`=LL. (Example: `SAVE_CRMASK`=Y; the default value is N.)

`SAVE_TRUE_ERROR` is used to output the squared error of each resultant as computed by the subroutine `makel1`, in the form of 32-bit float data-cube FITS files of $4224 \times 4096 \times N_{\rm res}$ pixels, one per SCA. The file extension is `_error_SCAXX.fits`, where XX is the SCA number. Squared errors account for all currently-implemented sources of error and noise. Allowed entries are "Y" for yes and "N" for no. (Example: `SAVE_TRUE_ERROR`=N, which is the default value.)

`SAVE_SATFLAG` allows the output of FITS files in which pixel values larger than zero indicate the first read in which pixels reached saturation. The file format is the same as for the CR masks. The extension is `_satflag_SCAXX.fits`, where XX is the SCA number. Valid entries are "Y" for yes and "N" for no. (Example: `SAVE_SATFLAG`=Y; the default value is N.)

[5]Note that Poisson noise in L1 data is, instead, treated carefully.

`SAVE_L2` allows the output of the calibrated L2 images in the form of 32-bit float 4088× 4088-pixel FITS files, one per SCA, obtained through ramp-fitting of calibrated L1 data. The output extension is `_L2_SCAXX.fits`, where XX is the SCA number. This parameter is only used when `MODE`=L2. (Example: `SAVE_L2`=Y; the default value is N.)

`SAVE_VARIANCE` is used to output the total variance of the up-the-ramp fitting used to create L2 images. The total variance is saved as 32-bit float FITS files of 4088×4088 pixels, one per SCA. The file extension is `_varnce_SCAXX.fits`, where XX is the SCA number. Valid entries are "Y" for yes and "N" for no. (Example: `SAVE_VARIANCE`=Y; the default value is N.)

`SAVE_META` allows the output of the astronomical scene on the meta frame (see Sect. 3.1 for details). SCAs are placed on the meta frame to the nearest integer pixel. The meta image is a 32-bit float, 26 630×16 540-pixel FITS file, with extension `_meta.fits`. Valid entries are "Y" for yes and "N" for no. (Example: `SAVE_META`=N, which is the default.)

2.2 Effective Point-Spread-Function models

This section is focused on how to construct input ePSF models for `WFIsim` and their FITS format.[6] The ePSF models closely follow the prescriptions given in Anderson & King (2000, 2006).

The ePSF is the convolution of the instrumental PSF $\psi_{\rm I}$ with the pixel-response function of the detector $\Pi_{\rm det}$:

$$\psi_{\rm E} = \psi_{\rm I} \otimes \Pi_{\rm det}. \tag{1}$$

The instrumental PSF is what the telescope produces at the focal plane, and can be approximated with the output of the Roman module of WebbPSF.[7] The pixel-response function encompasses several contributions, e.g.: intra-pixel sensitivity variations, inter-pixel capacitance (IPC), charge diffusion, brighter-fatter effects, etc. At present, `WFIsim` assumes that the only contribution to $\Pi_{\rm det}$ is the IPC. The values of the adopted 3×3-pixel IPC kernel are shown in Fig. 2.

The ePSF is a continuous function of the offsets from the center of the ePSF itself, whose value at any point is the fraction of light of a point source that would fall in a pixel centered at that point. The ePSF is generally much easier to use than any other representation of the PSF, since it requires no integration[8], and ePSF models of undersampled detectors are

[6]The authors can provide these ePSF models, or python and `Fortran` tools to construct them, upon request.

[7]`https://www.stsci.edu/jwst/science-planning/proposal-planning-toolbox/psf-simulation-tool`.

[8]One key aspect of the ePSF formalism is that it directly accounts for the integration of the PSF profile

0.21%	1.66%	0.22%
1.88%	91.59%	1.87%
0.21%	1.62%	0.20%

Figure 2: The 3×3-pixel grid of the IPC kernel. It shows how the flux in a pixel is lowered to 91.59% of its otherwise IPC-free value, and the removed flux is redistributed across its eight surrounding pixels.

usually supersampled to minimize systematic positional errors (see, e.g., Anderson & King 2000). It is often the case that the ePSF is empirically built from empirical data but, for now, we can construct ePSF models following the definition given in Eq. 1, using WebbPSF outputs as the instrumental PSF, and the IPC as the pixel-response function.

Because of the large field of view of the Roman WFI, very bright stars will likely be present in some SCAs of any given exposure. These sources will still carry significant signal along the PSF spikes hundreds of pixels away from their centers. To account for these effects, `WFIsim` makes use of three sets of ePSF models of increasing size: (i) the first set (the standard set) is represented by a grid of 3×3, $\times 4$-supersampled, 32-bit float models per SCA, and account for spatial variations across the focal plane. The models map the local ePSF at the four corners, the mid sides, and the center of each SCA, and extend out to 45 SCA pixels. (ii) The second set (the wide set) consists of a single, $\times 4$-supersampled 32-bit float model per SCA, extending out to 455 SCA pixels. This model is not spatially-varying within an SCA. It is only used for the wings and spikes of very bright sources, where spatial variability within an SCA is less relevant, since most of the variation signal is in the ePSF cores. (iii) Finally, the third set (the extra-wide set) is made up of a single, non-supersampled, 64-bit float ePSF model per SCA, extending out to 1001 SCA pixels in each direction. This last model is only used for those extremely bright stars whose signal along the spikes is still relevant over 455 pixels away from their centers. Note that, due to numerical round offs of the fast Fourier transforms in WebbPSF, the extra-wide sets show artificial spike-like features pointing inwards from the edges of the ePSF array. To minimize this unwanted feature, `WFIsim` limits the use of the extra-wide sets out to 750 pixels.

Following Anderson & King (2000, 2006) conventions, `WFIsim` requires ePSF models to be centered at the center of a pixel grid. This ensures a sample point at the extremum

over the pixel grid, therefore drastically simplifying usage and improving performance.

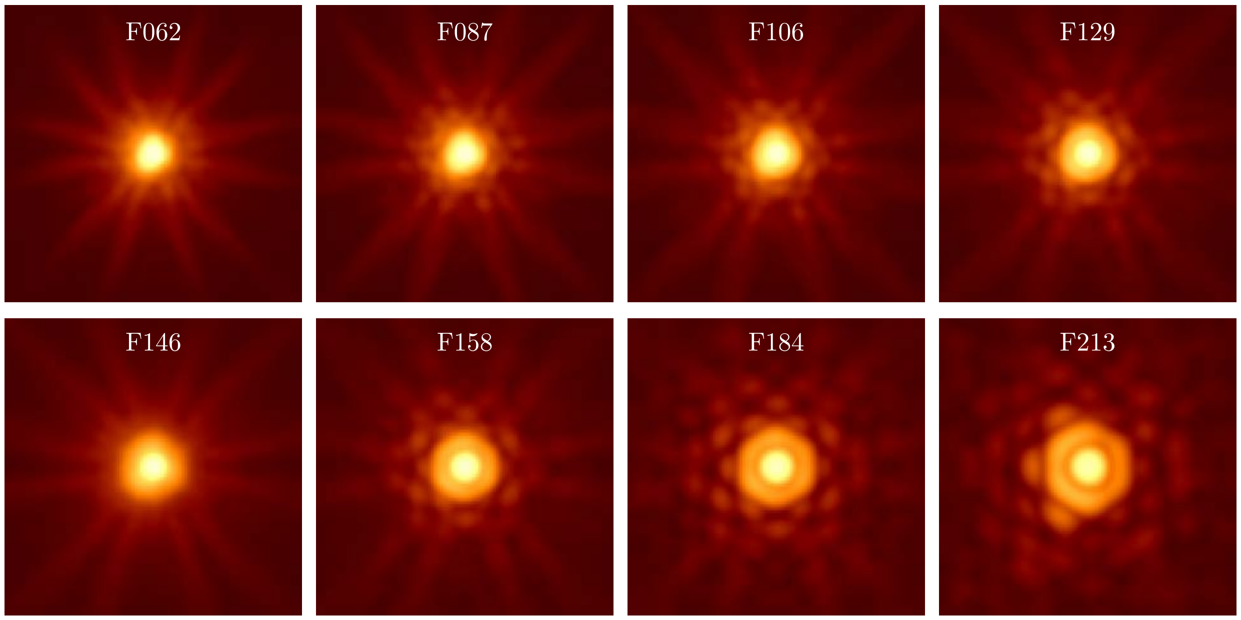


Figure 3: The central 101×101, $\times 4$-supersampled pixels of the ePSF models at the center of SCA 05 for the eight optical imaging elements of the WFI. These models extend out to 12.5 SCA pixels in each direction. The logarithmic color bar is the same in all panels.

of the ePSF, where it is maximum, rather than having the maximum to occur implicitly between four sample points. It also provides an easy-to-inspect metric of the central intensity of the ePSF. The current version of WebbPSF, as is, does not allow the output of even supersampling PSFs on an odd-size pixel grid using conventional parameters. However, a current work-around is to set the PSF-model sampling to 1, and to instead supersample the SCA pixels by a factor of four.

With the WebbPSF outputs in hand, the next step is to integrate them over the SCA pixel grid so that the derived ePSFs will satisfy the condition that their values at any point represent the fraction of light of a point source that would fall in a pixel centered at those points (this step is ignored for for extra-wide, non-supersampled models). Finally, we convolve the integrated PSFs with the pixel-response function $\Pi_{\rm det}$ (= IPC) of the WFI.

All the three sets of ePSF models are saved into single FITS files, one per SCA. The standard 3×3 models of each SCA are abutted into a single FITS file of size 1081×1081 pixels, in which the nine ePSFs are centered at X(Y) pixel positions 181, 541, 901. Figure 3 shows the core of central models of the standard ePSF set of SCA 05 for the eight optical imaging elements (filters) of the WFI. The same logarithmic colorbar is used in each panel. It is interesting to note that the F146 model is blurrier than, e.g., the F129 and F158 models, due to the F146 filter having a much wider bandpass. It is also worth noting that the bluer the filter, the tighter the ePSFs are, following the λ/D angular-resolution formula.

Figure 4 illustrates the differences between the core of the 3×3 spatially-varying F146

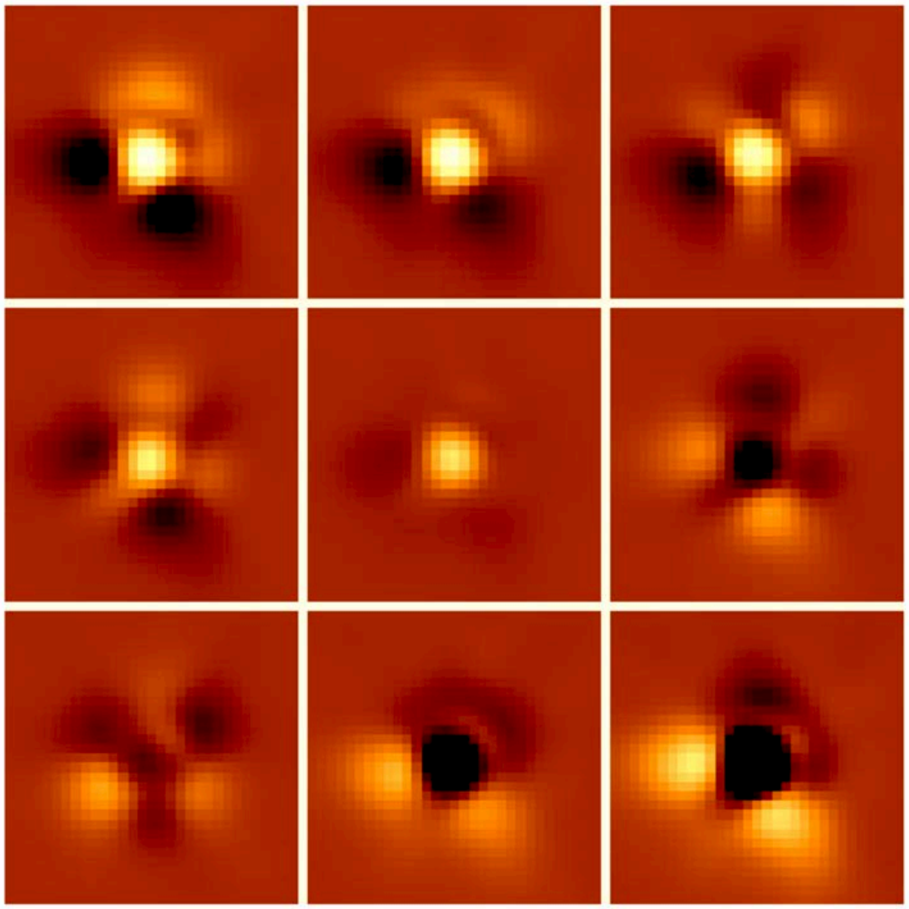


Figure 4: Illustration of the spatial variation of the ePSF models. The panels show the central 31×31 $\times 4$-supersampled pixels of the 3×3 grid of standard models of SCA 18 and filter F146. The logarithmic color bar is the same in all panels. The maximum peak-to-peak variation is around 3%.

ePSFs of the standard set of SCA18 and the average ePSF over the entire SCA. The maximum peak-to-peak variation is around 3%. Figure 5 compares the extension of the three ePSF sets of SCA 10 for the F146 filter. From left to right: the central ePSF of the standard set, the wide set, and the extra-wide set. The logarithmic colorbar in the three plots is individually adjusted to enhance ePSF features at different radial distances.

When `ADD_SCENE`=Y, `WFIsim` reads in the ePSF models from the `PSFPATH` directory. It expects a subdirectory with the filter name, containing all three sets of models. The models themselves must be named as: SCAXX_psflibipc.fits, SCAXX_widepsflibipc.fits, and SCAXX_xwidepsflibipc.fits, for the standard, wide and extra-wide sets, respectively, and where XX is the SCA number from 01 to 18. (Note the presence of an underscore after SCAXX for the standard model, and a dot for the wide and extra-wide models.)

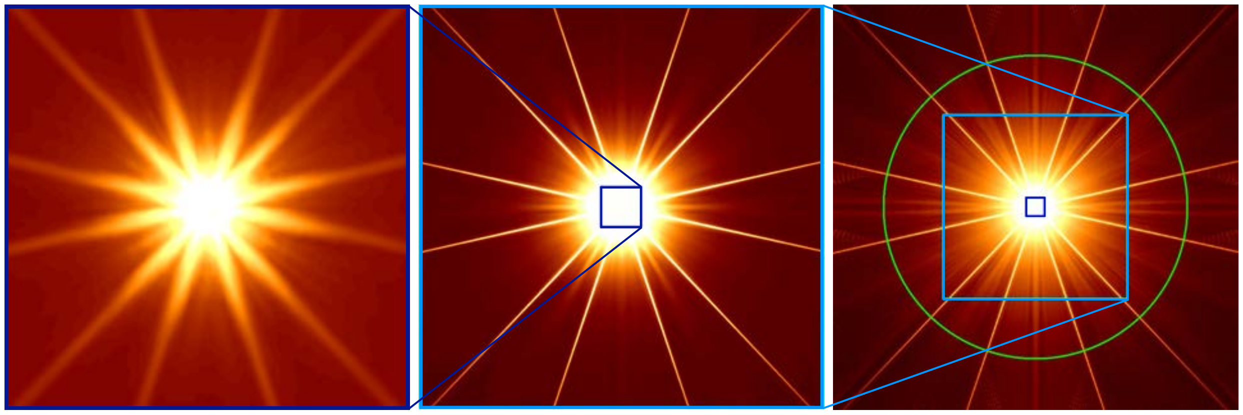


Figure 5: From left to right, the central F146 ePSF model of the standard set of SCA 10, the wide-set ePSF for the same SCA and filter, and the corresponding extra-wide-set ePSF. To better highlight fine features in the ePSF wings and spikes, the logarithmic colorbar of each panel is adjusted independently. The standard set covers 90×90 SCA pixels and is represented by an open blue square in the middle and right panels. Similarly, the wide set covers 910×910 SCA pixels and is represented by a light-blue open square in the right panel. The extra-wide set covers up to 2001×2001 SCA pixels, but the region outside the green circle of radius 750 pixels is currently not used in `WFIsim`. This is to minimize fast-Fourier-transform round offs in WebbPSF, which show up as dashed spikes diverging inwards from the mid points of the edges of the image.

3 Example run and program execution

Let us suppose that we want to simulate both level-1 and level-2 full-frame images of a field containing stars and galaxies, and that we have already prepared the necessary input source catalogs in sky coordinates, centered on (R.A., Dec.)=(80.0, 30.0). The catalogs cover two square degrees (in case we want to apply large dithers). The star catalog (stars.cat) contains around 15.5 million sources of various luminosities with a flat spatial distribution, plus half a million sources with a distribution mimicking that of a large globular cluster in the Milky Way, slightly offset at location (R.A., Dec.)=(80.21, 29.88). The galaxy catalog (galaxies.cat) contains half a million sources with a flat spatial distribution and Sérsic profiles resembling galaxies in the GOODS fields (Giavalisco et al. 2004).

In `IN.WFIsim` we set the following parameters: `MODE`=L2, `SCA_NUMBER`=18, `ADD_SCENE`=Y, `STARCAT_NAME`=/user/username/stars.cat, `GALCAT_NAME`=/user/username/galaxies.cat, `ADD_STARS`=Y, `ADD_GALAXIES`=Y, `NSTARS`=−1, `NGALS`=−1. We complete the astronomical scene by adding 1.5 $e^{-}\,s^{-1}$ of flat sky background, and instruct `WFIsim` to ignore all sources whose central pixel is below 1σ of the sky noise: `ADD_BACKGROUND`=Y, `BACKGROUND`=1.5, `FAINT_SIGMA_THRESHOLD`=1.0.

The pointing setup is as follows. Since positions in the input source catalogs are in sky coordinates, we set `POINTING_MODE`=RD. We place the nominal WFI aperture (to recall, this is defined as the center of the focal-plane coordinate system) at location (R.A., Dec.)=(79.965, 30.031) using `MRREF`=79.965 and `MDREF`=30.031, so that the center of the star cluster will end up within SCA 14. We also add a small pointing uncertainty of 0.01 arcsec (about a tenth of a pixel) with `ADD_DITHER_NOISE`=Y and `DITHER_NOISE_VALUE`=0.01, apply a roll angle of 8 degrees around the aperture (`ROLL_ANGLE`=8.0), and include jitter effects (`ADD_JITTER`=Y).

We simulate an exposure with the F146 filter of duration close to 200 seconds (note that the exact exposure time must be a multiple of the exposure time of a single read). To improve the dynamical range, we define a multi-accum table with uneven resultants, with the first resultants made by fewer reads. We also include a reference read at the beginning of the multi-accum table and a few skipped reads before the start of the last three resultants: `MATABLE`=ITTRTRTRRRTRRRTRRRRRRRRTRRRRRRRRTSSSRRRRRRRRTSSSRRRR-RRRRTSSSRRRRRRRRT. The `MATABLE` includes 68 reads and 11 resultants, for a total exposure time of 206.72 seconds. The effective exposure time can be calculated as the reference-read-subtracted mid point of the reads in the last resultant: 191.52 seconds.

To improve the realism of the simulation, we include CR events (`ADD_CR`=Y) and make sure all currently-implemented calibration files are used (`USEFLAT`=Y, `USEDARK`=Y, `USERN`=Y, `USEBIAS`=Y, `USEGAIN`=Y, `USECNL`=Y, `USEDQ`=Y, `USESAT`=Y). Finally, we turn on all output options and leave the remaining parameters to their default value: `SAVE_INPUT_POSITIONS`=Y, `SAVE_JITTER_KERNEL`=Y, `SAVE_L1`=Y, `SAVE_CRMASK`=Y, `SAVE_TRUE_ERROR`=Y, `SAVE_L2`=Y, `SAVE_SATFLAG`=Y, `SAVE_VARIANCE`=Y, `SAVE_META`=Y. For completeness, we list in Appendix A the input configuration file described here.

We run the program with the following shell command:

```
> ./WFIsim.e IN.WFIsim
```

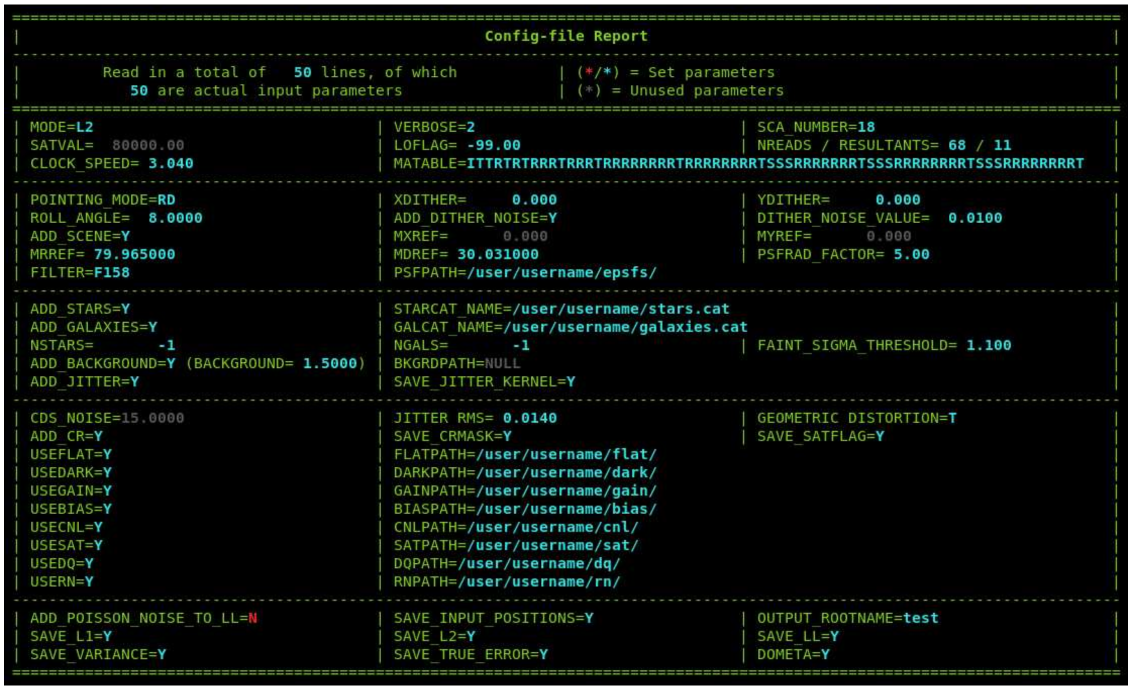
`WFIsim` parses the configuration file, sets up all parameters and prints a summary report on screen (see Fig. 6) that can be conveniently used to verify the full set up of the simulation.

3.1 Creation of the astronomical scene

The next step involves the construction of the astronomical scene as seen by the 18 SCAs (`SCA_NUMBER`=18). The astronomical scene is internally defined as a 32-bit float $4088 \times 4088 \times 18$ array, and contains all the input sources that fall within the SCAs given their coordinates, telescope pointing, geometric distortion, and that satisfy the minimum brightness threshold. The scene is in units of $e^- s^{-1}$, and is constructed by the subroutine `scenemaker`.

3.1.1 Coordinate transformations

Dithers and roll angles are treated as rotations around Euler angles on the celestial sphere. In this example run, input source positions are in sky coordinates, and thus Euler angle



```
====================================================================================================================
|                                              Config-file Report                                                  |
--------------------------------------------------------------------------------------------------------------------
|         Read in a total of   50 lines, of which            | (*/*) = Set parameters                              |
|            50 are actual input parameters                  | (*) = Unused parameters                             |
====================================================================================================================
| MODE=L2                                | VERBOSE=2                              | SCA_NUMBER=18                    |
| SATVAL=  80000.00                      | LOFLAG= -99.00                         | NREADS / RESULTANTS= 68 / 11     |
| CLOCK_SPEED= 3.040                     | MATABLE=ITTRTRTRRRTRRRTRRRRRRRRTRRRRRRRRTSSSRRRRRRRRTSSSRRRRRRRRTSSSRRRRRRRRT  |
--------------------------------------------------------------------------------------------------------------------
| POINTING_MODE=RD                       | XDITHER=     0.000                     | YDITHER=     0.000               |
| ROLL_ANGLE=  8.0000                    | ADD_DITHER_NOISE=Y                     | DITHER_NOISE_VALUE=  0.0100      |
| ADD_SCENE=Y                            | MXREF=       0.000                     | MYREF=       0.000               |
| MRREF= 79.965000                       | MDREF= 30.031000                       | PSFRAD_FACTOR= 5.00              |
| FILTER=F158                            | PSFPATH=/user/username/epsfs/                                             |
--------------------------------------------------------------------------------------------------------------------
| ADD_STARS=Y                            | STARCAT_NAME=/user/username/stars.cat                                     |
| ADD_GALAXIES=Y                         | GALCAT_NAME=/user/username/galaxies.cat                                   |
| NSTARS=        -1                      | NGALS=        -1                       | FAINT_SIGMA_THRESHOLD= 1.100     |
| ADD_BACKGROUND=Y (BACKGROUND= 1.5000)  | BKGRDPATH=NULL                                                            |
| ADD_JITTER=Y                           | SAVE_JITTER_KERNEL=Y                                                      |
--------------------------------------------------------------------------------------------------------------------
| CDS_NOISE=15.0000                      | JITTER RMS= 0.0140                     | GEOMETRIC DISTORTION=T           |
| ADD_CR=Y                               | SAVE_CRMASK=Y                          | SAVE_SATFLAG=Y                   |
| USEFLAT=Y                              | FLATPATH=/user/username/flat/                                             |
| USEDARK=Y                              | DARKPATH=/user/username/dark/                                             |
| USEGAIN=Y                              | GAINPATH=/user/username/gain/                                             |
| USEBIAS=Y                              | BIASPATH=/user/username/bias/                                             |
| USECNL=Y                               | CNLPATH=/user/username/cnl/                                               |
| USESAT=Y                               | SATPATH=/user/username/sat/                                               |
| USEDQ=Y                                | DQPATH=/user/username/dq/                                                 |
| USERN=Y                                | RNPATH=/user/username/rn/                                                 |
--------------------------------------------------------------------------------------------------------------------
| ADD_POISSON_NOISE_TO_LL=N              | SAVE_INPUT_POSITIONS=Y                 | OUTPUT_ROOTNAME=test             |
| SAVE_L1=Y                              | SAVE_L2=Y                              | SAVE_LL=Y                        |
| SAVE_VARIANCE=Y                        | SAVE_TRUE_ERROR=Y                      | DOMETA=Y                         |
====================================================================================================================
```

Figure 6: Summary table that `WFIsim` prints out on screen at the beginning of the run. The table lists all configuration parameters and their values that will be used in the simulation, and provides a convenient way to verify that the input configuration file has been correctly read.

rotations can immediately be applied to them. When input positions are given in pixel units on a tangent plane, deprojection of the plane on the celestial sphere around the aperture is computed prior to applying dithers and roll angles.

Next, the positions are projected on to a tangent plane around the location of the nominal aperture (`MRREF`=79.965 and `MDREF`=30.031 degrees) and converted to pixels units using a pixel scale of 110 mas pixel^{-1}. Finally, `scenemaker` applies the inverse of the geometric-distortion correction to obtain raw coordinates (see below). At this point, the origin of the coordinate system corresponds to the location of the aperture: we shift it to position (X, Y)=(13 315, 10 225) so that sources within the SCAs always map to positive pixels. We call this final coordinate system the raw *meta frame*. The raw SCA and meta positions of sources landing within each SCA are saved into two log files (`SAVE_INPUT_POSITIONS`=Y), one for stars and one for galaxies. When `VERBOSE`=2, `scenemaker` reports on screen how many stars and galaxies from the input catalogs have been added to each SCA, and how many of them were too faint to be considered.

A note must be made about the current distortion solution (and its inverse) in `WFIsim`. The distortion model follows cycle-6 specifications, and it was initially implemented to study the astrometric capabilities of the WFI in 2018. The model is now outdated and does not match that of the most recent Roman Science Instrument Aperture File (SIAF). We plan to update the distortion model of `WFIsim` in the next release of the code. There is no `IN.WFIsim` parameter that can turn distortion on or off, but this can still be done by changing the logical variable `_gd_` in the preamble of the source code and recompiling it. More details on the current distortion model and on the auto-calibration approach of the distortion solution will be provided in a separate technical report (Bellini in prep.). Here it suffices to say that the overall distortion of the WFI is of the order of 1%, meaning that the relative position of stars at the opposite corners of the WFI can be off by around 300 pixels (∼35 arcsec).

3.1.2 The scene array

The astronomical scene is constructed with the three user-defined types of sources: sky background, stars and galaxies. The background is added first. If it is provided through FITS files, it will also be convolved with the IPC kernel. For stars, `scenemaker` calculates the best ePSF model at their central pixel location through a bi-linear interpolation of the spatially-varying ePSF library models. A bi-cubic interpolation of this best ePSF model is then used to estimate a star flux in each pixel given its subpixel offset.

To improve performance, the flux of a star is added to the scene array only out to a certain radius, r_*, after which the flux contribution to the pixels of the scene falls well below the sky background noise. This radius is calculated by comparing the ePSF radial profile, scaled by the total flux of the star and `PSFRAD_FACTOR`, to `FAINT_SIGMA_THRESHOLD` times the sky background noise at the end of the exposure. The radius is required to be in the range $5 \leq r_* \leq 750$. When $r_* > 45$ pixels, flux is added using the wide ePSF models, and when $r_* > 455$ the extra-wide models are used.

Galaxies are also added to the scene array out to a certain distance from their centers that depends on their radial profile compared to the sky background noise. This is also done to improve performance. In addition, to better account for the fact that a galaxy profile can vary rapidly within its centermost pixels, `scenemaker` applies a two-step rectangular rule integration and supersamples the central 5×5 pixels by a factor of 11, and then by a factor of 5 out to $2\,R_\mathrm{e}$ where possible. The galaxy flux $F_{i,j}$ of pixel i,j is modeled given the input parameters and the following equations according to Ciotti (1991) and Peng (2002):

$$
\begin{aligned}
&i_\mathrm{rot} = (i - x_\mathrm{gal})\cos\theta - (j - y_\mathrm{gal})\sin\theta + x_\mathrm{gal}\\
&j_\mathrm{rot} = (i - x_\mathrm{gal})\sin\theta + (j - y_\mathrm{gal})\cos\theta + y_\mathrm{gal}\\
&r = \sqrt{(i_\mathrm{rot} - x_\mathrm{gal})^2 + \left(\tfrac{j_\mathrm{rot} - y_\mathrm{gal}}{1-\mathrm{Ell.}}\right)^2}\\
&f = 10^{-0.4\cdot\mathrm{Instr.\ mag}}\\
&b_n = 2n - \tfrac{1}{3} + \tfrac{4}{405n} + \tfrac{46}{25515n^2}\\
&I_e = \tfrac{f b_n^{2n}}{2\pi n R_e^2}\tfrac{\exp(-b_n)}{\Gamma(2n)}\tfrac{1}{1-\mathrm{Ell.}}\\
&F_{i,j} = I_e \exp\left(-b_n\left(\tfrac{r}{R_e}\right)^{\frac{1}{n}} - 1\right),
\end{aligned}
\tag{2}
$$

where θ is the inclination angle of the galaxy, in radians, ($x_{\rm gal}$, $y_{\rm gal}$) are the raw SCA pixel coordinates of the galaxy, "Ell." is the ellipticity, f is the instrumental total flux, b_n is approximated to the forth term, and I_e is the galaxy flux *at* the effective radius. All other quantities have been defined in Sect. 2.1.

After a model is constructed, it is convolved with the ePSF appropriate for its location on the SCA and is added to the scene array. At this time, only the standard ePSF models are used for the convolution, since the core of most galaxies are not bright enough to require the use of the wider ePSF models. An extension of the convolution kernel to include wider ePSF models is under consideration for future versions of `WFIsim`. Finally, and again to improve performance, the size of the convolution kernel across the image is not constant: it is largest for the central galaxy pixels, where most of the flux is, of average size at intermediate distances, and is smallest elsewhere. The exact convolution kernel sizes and to which pixels they should be applied can be defined in the preamble of the source code through the three `_gal_conv_radiusX_` parameters, where X is 0, 1 or 2 for the central, intermediate, and farther out pixels, and the two `_gal_conv_mask_radiusX_`, where X is either 1 or 2, and define the boundaries between central and intermediate regions, and between intermediate and farther-out regions. For the highest fidelity, we recommend to set all `_gal_conv_radiusX_` parameters to the maximum value of 43, since only the standard ePSF models have been currently implemented for the convolution. (For comparison, the default convolution window in Galfit (Peng 2002) has a square radius of 50 pixels.)

Jitter effects are rendered as a 2D Gaussian kernel with semi-major and semi-minor axis values randomly drawn from a Gaussian distribution with $\sigma = 14$ mas (a somewhat conservative estimate of the expected jitter RMS for Roman[9]), and a random inclination between 0 and 180 degrees, with a zero-degree inclination implying that the major axis of the jitter kernel is parallel to the X axis of the meta frame. The scene array is then convolved with the jitter kernel. A $\times 10$-supersampled version of the inner 3×3 pixels of the jitter kernel can be saved to disk as FITS file (`SAVE_JITTER_KERNEL`=Y). The supersampled jitter kernel of the run is shown in Fig. 8.

The user has the option to output the astronomical scene as FITS files for both the individual SCAs (`SAVE_LL`=Y) and as single file for the full-frame WFI (`SAVE_META`=Y), as shown in Fig 7. In the figure, SCA numbers are reported near the top-right corner of each SCA. The nominal aperture of the WFI is marked with a red cross, and lies in the gap between SCA 10 and SCA 01. Zoomed-in views of a subregion of SCA 15 and SCA 07, characterized by significantly different source distributions, are shown in the bottom panels.

[9]But note that the value of the jitter RMS can be changed in the preamble of the source code, see Sect. 2.1.

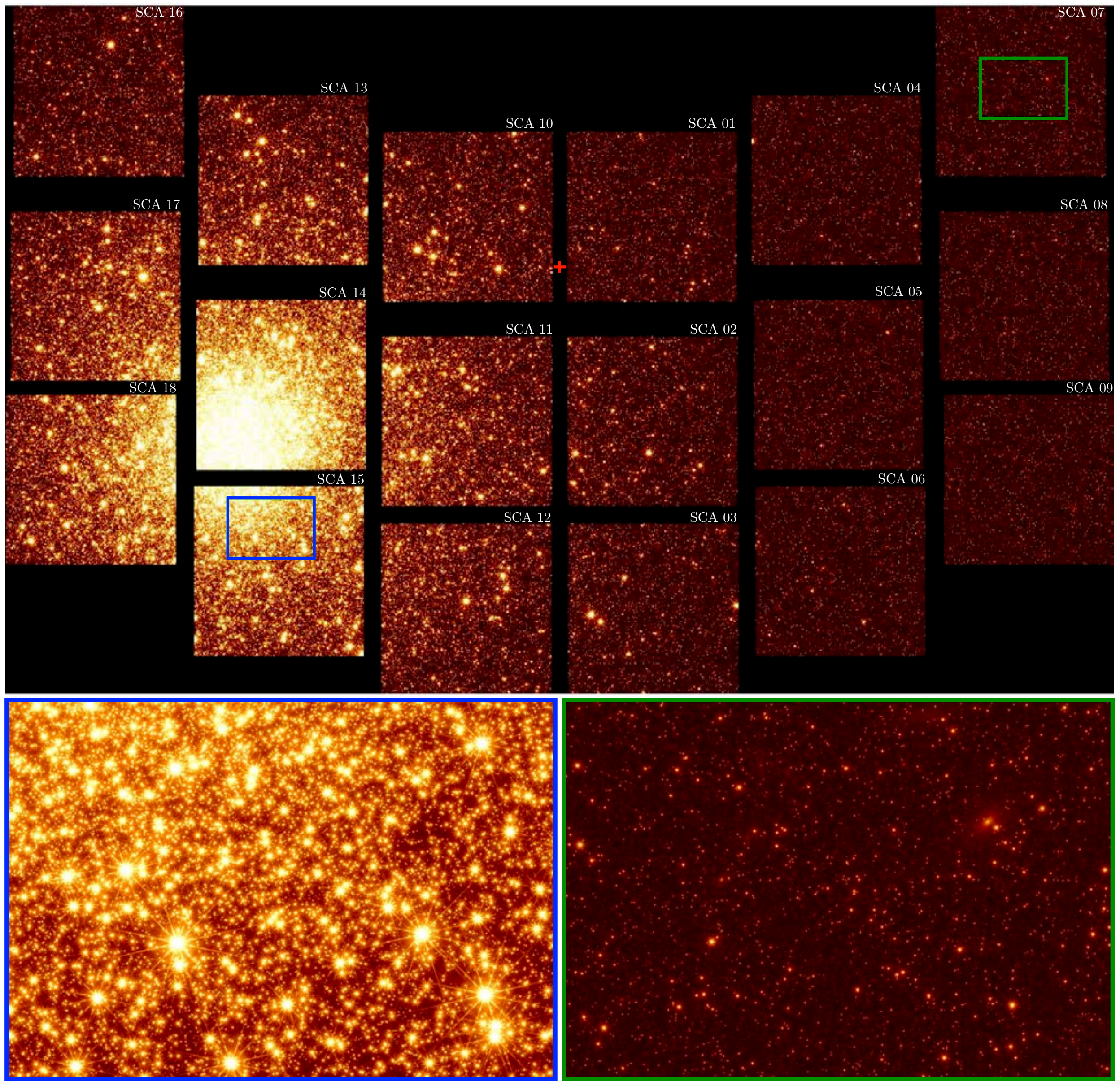


Figure 7: The top panel shows the full-view scene as seen on the meta frame (extension: `_meta.fits`). The image size is 26630 × 16540 pixels, leaving a 1-pixel-wide border around the edges. SCA numbers are listed on the top-right corner of each SCA, for clarity. The red cross near the center of the frame is the location of the nominal aperture of the WFI, at pixel (x, y)=(13315, 10225) on the meta frame. The star cluster is centered at the bottom of SCA 14. Two regions are zoomed-in in the bottom panels: on the left a region of SCA 15 near the center of the cluster (highlighted in blue in the top panel), with many bright stars whose spikes extends for hundreds of pixels; on the right a region of SCA 07 (highlighted in green in the top panel) where a few bright galaxies can be easily seen thanks to the lower stellar density.

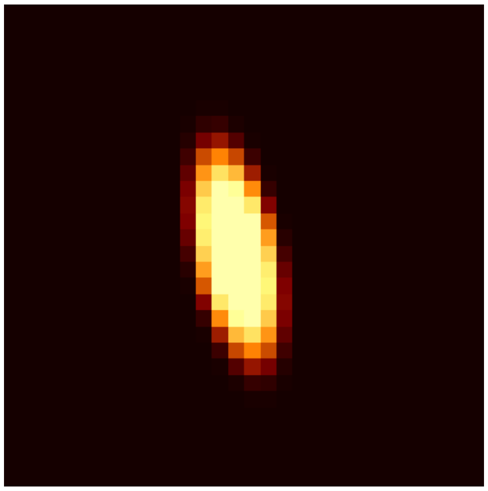


Figure 8: The 30×30 pixels of the $\times 10$-supersampled jitter kernel of the run. The 2D Gaussian has major and minor axes of 0.185 and 0.069 pixels, respectively (or 20.35 and 7.59 mas), while the inclination angle of the major axis (counterclockwise from the X axis) is 101.165 degrees.

3.2 Creation of level-1 images

Level-1 images (L1) contain the raw pixel data of the WFI in the form of resultant frames, which consist of a set of reads averaged together according to a multi-accum table. The L1 data are stored into $4224 \times 4096 \times N_{\mathrm{res}}$ pixels, where N_{res} is the number of resultants in the exposure. Each SCA is read by 33 amplifiers: 32 of them read 128 physical pixel columns (for a total of 4096 columns) and the 33rd amplifier, clocked in the same way as the others, reads 128 virtual pixel columns (the reference pixels) that can be used for calibration purposes. In addition, only the inner 4088×4088-pixel region of the 4096×4096 physical pixels of a SCA is exposed to the astronomical scene.[10] Currently, `WFIsim` L1 images contain the full $4224 \times 4096 \times N_{\mathrm{res}}$ pixel space.

A specific subroutine, `makel1`, is designed to construct L1 data through the simulation of all the reads of a multi-accum table. Five main arrays and two auxiliary arrays are used in the process, all with the same size as the L1 image:

Main arrays:

- The *read scene* (32-bit float) records the total amount of signal (in electrons) that is ideally present in a single read in an SCA. The total signal is the sum of the astronomical scene defined in Sect. 3.1, scaled by the inverse of the flat plus the dark current rate. The sum is multiplied by 3.04 seconds (`CLOCK_SPEED`) to match the exposure time of a single read.

[10]Note that, at the time of writing, it is still unclear whether information from the 33rd amplifier and/or the four-pixel-wide borders of the unexposed SCAs will be left within the Roman ADSF L1 images, or will be trimmed out and stored into a separate ADSF file.

- The *accumulator* (32-bit integer) keeps track of the increasing amount of electrons present in the SCAs as the exposure progresses. It is updated every read.
- When the first character of the `MATABLE` is "I", the first read is treated as the reference read, and the pixel values in the accumulator are read in (hence, read-noise is added), converted into counts, and stored into the *reference-buffer* (16-bit integer) array after the bias is also added.
- For all the subsequent reads that are not skipped, the accumulator is also read in, and pixel values are converted into counts and added to the *main buffer* array (32-bit integer) together with the bias.
- When it is time to assemble a resultant frame (`MATABLE`=T), the main buffer is divided by the number of reads that were added to it, the reference buffer is subtracted and the bias is added, and the result is stored into the *L1* array (16-bit integer). At the end of the process, the main buffer is zeroed out to be ready for the next set of reads.

Auxiliary arrays:

- The *saturation-flag* array (16-bit integer) records the first read in which a pixel in the accumulator reaches saturation.
- The *CR mask* array (16-bit integer) keeps track of the first read in which pixels of the accumulator are hit by a CR. Note that only the first CR event in a pixel is recorded in the CR mask, but pixels in the accumulator can potentially be hit by many CRs during an exposure.

Note that the bias is always added when read noise is involved, to avoid integer underflow issues in the buffer and L1 arrays. In what follows, we provide a more in-depth, step-by-step description of the tasks performed by the subroutine `makel1` during an exposure.

`makel1` starts by loading all user-defined calibration files (a few examples are shown in Fig. 9) and by setting up the read-scene array. Then for each read, the read-scene array is added to the accumulator, together with shot noise. Based on theoretical considerations, photons hitting the WFI should be treated as a random variable that is drawn from a Poisson distribution, however this can be computationally expensive. Therefore, we leverage the property that a Poisson distribution approaches a Gaussian one for large values of $S_{\rm CNL}$, the CNL-affected signal, with a variance given by $\sigma^2 = S_{\rm CNL}$. The subroutine `makel1` makes use of the exact Poisson form for pixels with a total added signal $S_{\rm CNL} < 200$ e$^-$ s^{-1} and the Gaussian approximation for $S_{\rm CNL} \geq 200$ e$^-$ s^{-1}. This assumption implies that pixels whose signal is dominated by background flux are generally treated as Poisson deviates, whereas pixels dominated by source light (i.e., stars or galaxies) are generally treated as Gaussian

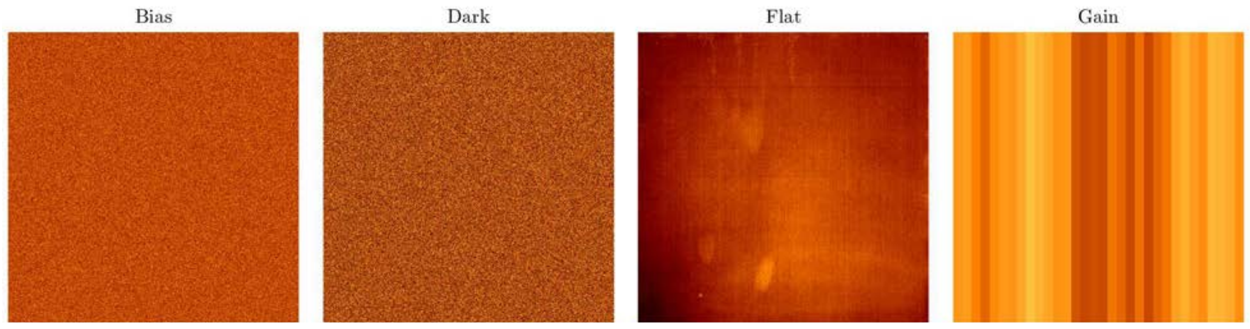


Figure 9: Example of calibration files for SCA 01 used in the simulation. From left to right: master bias, master dark rate, master flat, and master gain files. The color bar is shown at the bottom of each panel. The color scales is linear in all cases.

deviates. Furthermore, since most pixels in many simulations will be sky-dominated, their noise will likely be treated as Poisson deviates.

When CNL calibration files are provided in input (`USECNL`=Y), the subroutine `makel1` computes and updates the five main arrays mentioned above for both the CNL-affected and -unaffected cases. The CNL-affected L1 array is outputted as L1 FITS files, while the CNL-unaffected L1 array is passed to the subroutine that computes level-2 images. This design choice provides a significant increase in performance during the creation of level-2 data, at the cost of a slight decrease in performance for `makel1`. In fact, the correction of CNL-affected L1 data would require a complicated, iterative procedure to reconstruct what the individual CNL-affected reads should be, correct them for CNL effects, and rebuild the resultant frames without having knowledge of what is stored in the reference read, since the reference read, to the best of our knowledge, will not be downlinked to ground. On the other hand, having at our disposal a CNL-free set of resultant frames will assure ideal construction of level-2 images. When `USECNL`=Y, only a fraction of the read-scene signal is actually added to the CNL-affected accumulator. This fraction is a function of the charge already present in the CNL-accumulator, and is estimated using the inverse CNL solution.

When `ADD_CR`=Y, cosmic-ray events are simulated for each SCA, convolved with the IPC and added to the accumulator for each read. The the auxiliary CR mask array records which pixel is affected by CRs for the first time. The final CR mask arrays can be saved as FITS files (`SAVE_CRMASK`=Y). As an example, the top panels of Fig. 10 show the CR mask of the central 300 $\times$ 300-pixel region of SCA 16 for resultants 1, 4, 8 and 11, from left to right, respectively. The linear-scaling color bar is the same in all four panels.

When a pixel in the accumulator reaches saturation, its value is kept at saturation level throughout the rest of the exposure, and the corresponding pixel in the auxiliary saturation-flag array is updated with the read number in which saturation is first reached. The final saturation flag array can also be saved as FITS files (`SAVE_SATFLAG`=Y). The second row of panels in Fig. 10 show the same region of SCA 16 for the same four resultants of the

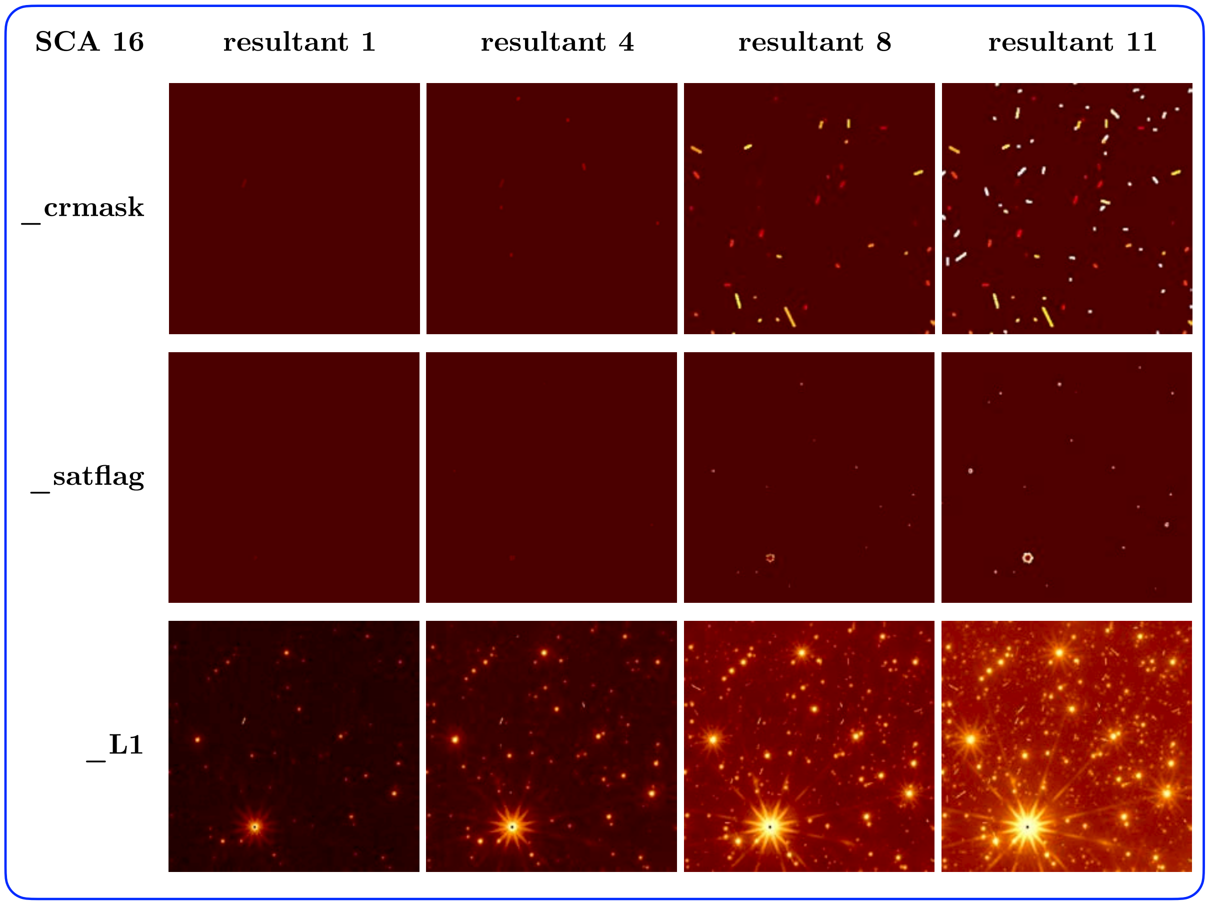


Figure 10: Example of data generated by the `makel1` subroutine for the central region (300×300 pixels) of SCA 16. From left to right we show resultants 1, 4, 8 and 11, respectively. Panels from top to bottom are for the CR mask, the saturation flag and the L1 resultants, respectively. Color bars are the same in each row of panels, for a direct comparison. The color scale is linear in the top two rows of panels, and logarithmic for the bottom row. (See the text for details.)

saturation-flag array. As for the CR mask panels, the same linear-scaling color bar is applied to the four panels of the saturation-flag array. Both auxiliary arrays will be used during the ramp-fitting process (more in Sect. 3.3).

When a read is not skipped (`MATABLE`≠S), the accumulator is then read and added to the main buffer. This process involves read noise, which is also added to the main buffer together with the bias. Since the main buffer stores pixel values in units of ADUs, the inverse gain correction is also applied:

$$\mathrm{MB}_{ijk} = \sum_{r=1}^{R_n} \left\lfloor \frac{\mathrm{A}_{ijk} + \mathrm{RN}_{ijk} + \mathrm{B}_{ijk}}{\mathrm{G}_{ijk}} \right\rfloor , \qquad (3)$$

where MB, A, RN, G and B are the main buffer, the accumulator, the read noise, the gain and the bias arrays, respectively. The sum is over the reads r in a set that will be used to assemble a resultant frame, with a total of R_n reads for resultant n. The subscript ijk indicates that the process is done for all pixels i, j of SCA k. While the main-buffer array is defined as a 32-bit integer in `makel1`, the on-board buffer of Roman actually operates at 24 bits, therefore the maximum value a pixel can reach in the main buffer is limited to 2^{24}, or about 16.8 million counts.

When the first read of the exposure is set as the reference read (`MATABLE`=I), the main buffer is copied to the reference buffer before and it is zeroed out to be ready for the next set of reads of a resultant frame.

When it is time to assemble a resultant frame (`MATABLE`=T), the accumulator is read and added to the main buffer one last time. The resultant frame is then obtained by dividing the main buffer by the total number of reads R_n that were added to it, subtracting the reference buffer, and adding the bias:

$$\mathrm{L1}^{(n)}_{ijk} = \left\lfloor \frac{\mathrm{MB}_{ijk}}{R_n} - \mathrm{RR}_{ijk} + \frac{\mathrm{B}_{ijk}}{\mathrm{G}_{ijk}} \right\rfloor , \tag{4}$$

where $\mathrm{L1}^{(n)}$ is the n-th resultant of the L1 image, and RR is the reference read. Finally, the main buffer is zeroed out again to be ready for the next set of reads of the next resultant frame. The bottom panels of Fig. 10 show the same central region of SCA 16 for the same resultants of the L1 image. The color bar is again the same in the four panels but this time the scale is logarithmic. Resultant frames appear increasingly bright as the exposure progresses from left to right. The number of CR traces also increase from left to right. The central pixels of the bright star near the bottom-left corner saturate in the very first read (the reference read), and their value in the resultant frames is similar to that of the bias, modulo read-noise variations.

3.3 Creation of level-2 images

The subroutine `makel2` is responsible for calibrating L1 images and performing ramp fitting. It outputs two 4088×4088-pixel, 32-bit float arrays per SCA: the L2 array contains the slope of the ramps in units of $\mathrm{e}^- \, \mathrm{s}^{-1}$, and the variance array contains the total variance of the fits. L2 and variance data can be saved as FITS files with `SAVE_L2`=Y and `SAVE_VARIANCE`=Y. The dark correction is currently not applied in `makel2`. The ramp fitting process is done following the prescriptions given in Casertano (2022). `makel2` first applies bias and gain corrections to the CNL-unaffected L1 data, then it goes through the fitting process, pixel-by-pixel, for the region of the L1 array exposed to the astronomical scene, that is, pixels [5:4092,5:4092].

Only unsaturated resultants are considered in the fit. When the only available information is in the L1 data, as it is the case for the Roman calibration pipeline, it is not trivial to identify within which resultant a pixel actually saturates. In `makel2`, the resultants in which pixels saturate are immediately identified using the saturation-flag array.

The currently adopted weighting scheme is a slightly modified version of that of the JWST NIRCAM calibration pipeline, to account for uneven resultants (Casertano 2022):

$$w_n = \frac{(1+P)R_n}{1+P\cdot R_n}\,|\bar{t}_n - \bar{t}_{\rm mid}|^P, \tag{5}$$

where w_n is the weight associated to resultant n, $\bar{t}_n$ is the effective time of the resultant, i.e., the mid-point of the exposure time of the reads associated to that resultant, $\bar{t}_{\rm mid} = (\bar{t}_N + \bar{t}_1)/2$ is the mid-point of the effective times of all resultants, and the exponent P is defined as:

$$P = \begin{cases} 0, & \text{if } s < 5. \\ 0.4, & \text{if } 5 \le s < 10. \\ 1, & \text{if } 10 \le s < 20. \\ 3, & \text{if } 20 \le s < 50. \\ 6, & \text{if } 50 \le s < 100. \\ 10, & \text{if } 100 \le s; \end{cases} \tag{6}$$

where the signal-to-noise parameter s is given by:

$$s = \frac{S_{\rm max}}{\sqrt{{\rm RN}^2 + S_{\rm max}}}, \tag{7}$$

with $S_{\rm max}$ being the value, in electrons, of the last resultant.

Following a least-squares approach and using the auxiliary quantities:

$$\begin{aligned} &\mathcal{F}0 = \textstyle\sum_{n=1}^N w_n \\ &\mathcal{F}1 = \textstyle\sum_{n=1}^N w_n \bar{t}_n \\ &\mathcal{F}2 = \textstyle\sum_{n=1}^N w_n \bar{t}_n^2 \\ &\mathcal{D} = \mathcal{F}2\cdot\mathcal{F}0 - \mathcal{F}1^2 \\ &\mathcal{K}_n = (\mathcal{F}0\cdot\bar{t}_n - \mathcal{F}1)\cdot\tfrac{w_n}{D}, \end{aligned} \tag{8}$$

the estimated slope $\hat{\mathcal{F}}$ is given by:

$$\hat{\mathcal{F}} = \sum_{n=1}^{N} \mathcal{K}_n \mathcal{S}_n, \tag{9}$$

where $\mathcal{S}_n$ is the value of a L1-calibrated pixel of resultant n.

`makel2` also computes the read-noise variance ν_{RN}, the signal variance ν_{S}, and the total variance ν_{tot}:

$$\begin{aligned}
&\nu_{\mathrm{RN}} = \textstyle\sum_{n=1}^{N} \mathcal{K}_n^2 \frac{\mathrm{RN}^2}{R_n} \\
&\nu_{\mathrm{S}} = \textstyle\sum_{n=1}^{N} \mathcal{K}_n^2 \tau_n + \sum_{n<m} 2\mathcal{K}_n\mathcal{K}_m \cdot \bar{t}_n \\
&\nu_{\mathrm{tot}} = \nu_{\mathrm{RN}} + \nu_{\mathrm{S}} \cdot \hat{\mathcal{F}},
\end{aligned} \tag{10}$$

where τ_n is the variance-based resultant time, defined as:

$$\tau_n = \sum_{r=1}^{R_n} [2(R_n - r) - 1] \cdot \frac{t_{n,r}}{R_n^2}. \tag{11}$$

We refer the interested reader to the Casertano (2022) report for a detailed description of the algorithm. Only the total variance ν_{tot} is currently saved into FITS files when `SAVE_VARIANCE`=Y. We plan to allow the user to also save the two components ν_{RN} and ν_{S} in future versions of `WFIsim`.

The `makel2` subroutine does not perform any jump-detection in order to identify resultant frames affected by CRs, since it takes direct advantage of the CR masks created by `makel1`. When a pixel is affected by a CR event, its ramp is split into 2 parts: a pre-event sub-ramp and post-event sub-ramp, and the resultant where the CR appears is excluded. The ramp fitting is performed on a sub-ramp if it contains at least three resultant values. `makel2` considers eight cases according to how many unsaturated resultant values are available in a sub-ramp. When a ramp is unaffected by CRs or saturation, it is considered as a pre-event ramp.

For six of the eight cases, the ramp-fitting algorithm is not used because there are less than three available resultant values in any given sub-ramp. When this happens, the slope value is approximated when possible, and specific flag values are used for the corresponding pixels of the variance array.

(1) There are zero resultants available, e.g., a pixel is already saturated in the first resultant: the corresponding pixel in the L2 array is set to `LOFLAG` and that of the variance array to -900.

(2) A pixel saturates in the second resultant: the slope is obtained as the value in the first resultant divided by the effective time of the first resultant: $\hat{\mathcal{F}} = \mathcal{S}_1/\bar{t}_1$. The corresponding pixel in the variance array is set to -800.

(3) Similar to Case (2), but the only available resultant value is in the second sub-ramp (e.g., a pixel is hit by a CR during the first resultant and its value is saturated in the forth resultant): the slope is still approximated similarly as for Case (2) but using values of the third resultant, and the corresponding variance pixel is set to -700.

(4) There are two available resultants before a CR event and less than two after: the slope

is approximated as the difference between the two resultant values, divided by the difference between the two effective exposure times: $\hat{\mathcal{F}} = (\mathcal{S}_2 - \mathcal{S}_1)/(\bar{t}_2 - \bar{t}_1)$. The variance flag in this case is -600.

(5) Similar to Case (4) but the two available resultants are in the second sub-ramp. The slope is calculated as in Case (4), using the two resultant values of the second sub-ramp. The corresponding variance flag is -500.

(6) There are two available resultants values in both sub-ramps. In this case, `makel2` will only consider the second sub-ramp and proceed as for Case (5), but it will assign the value -400 to the corresponding pixel in the variance array.

The remaining two cases deal with at least three resultant values and make use of the ramp-fitting step:

(7) There are at least three resultant values in one sub-ramp ramp and less than three in the other sub-ramp. The sub-ramp with the most resultants is fitted using Casertano (2022) algorithm: the resulting slope $\hat{\mathcal{F}}$ is stored into the L2 array and the total variance $\nu_{\rm tot}$ into the variance array.

(8) Both sub-ramps contain at least three valid resultant values. In this case, `makel2` fits the two sub-ramps independently and combines the results together. The final slope is the weighted sum of the two slopes, obtained using the inverse of the two sub-ramp-based total variances as weights, and the final variance is estimated using the combined slope value.

At the end of the ramp-fitting process, `makel2` applies the flat-field correction to both the rate and the variance arrays. Finally L2 and variance arrays are checked against the DQ array, and bad pixels are flagged with the value `LOFLAG` in both arrays.

The top panels of Fig. 11 show the first and last resultants (left and right, respectively) of a 400×400-pixel region of SCA 04. The brighter and fainter bands of pixels, clearly visible on the top-left panel and to a lesser extent on the top-right panel, are due to slightly different gain values of the associated read-out amplifiers. The total variance array of the same region of SCA 04 is shown in the bottom-left panel, and that of the L2 array is in the bottom-right panel. Pixels with higher signals have higher variance, following Eq. 10. Pixels affected by CRs also have higher variance, due to the fewer available resultants used to compute the slope. Two stars are marked with open circles. The central pixel of the star in the light-blue circle saturated during the fourth read (third resultant). Its flux is approximated according to Case (4) and the variance value is set to -600. The central four pixels of the star encircled in dark blue, on the other hand, saturated during the first read and, following Case (1), the corresponding pixels in the L2 image are at `LOFLAG` value, and these of the total variance are at -900.

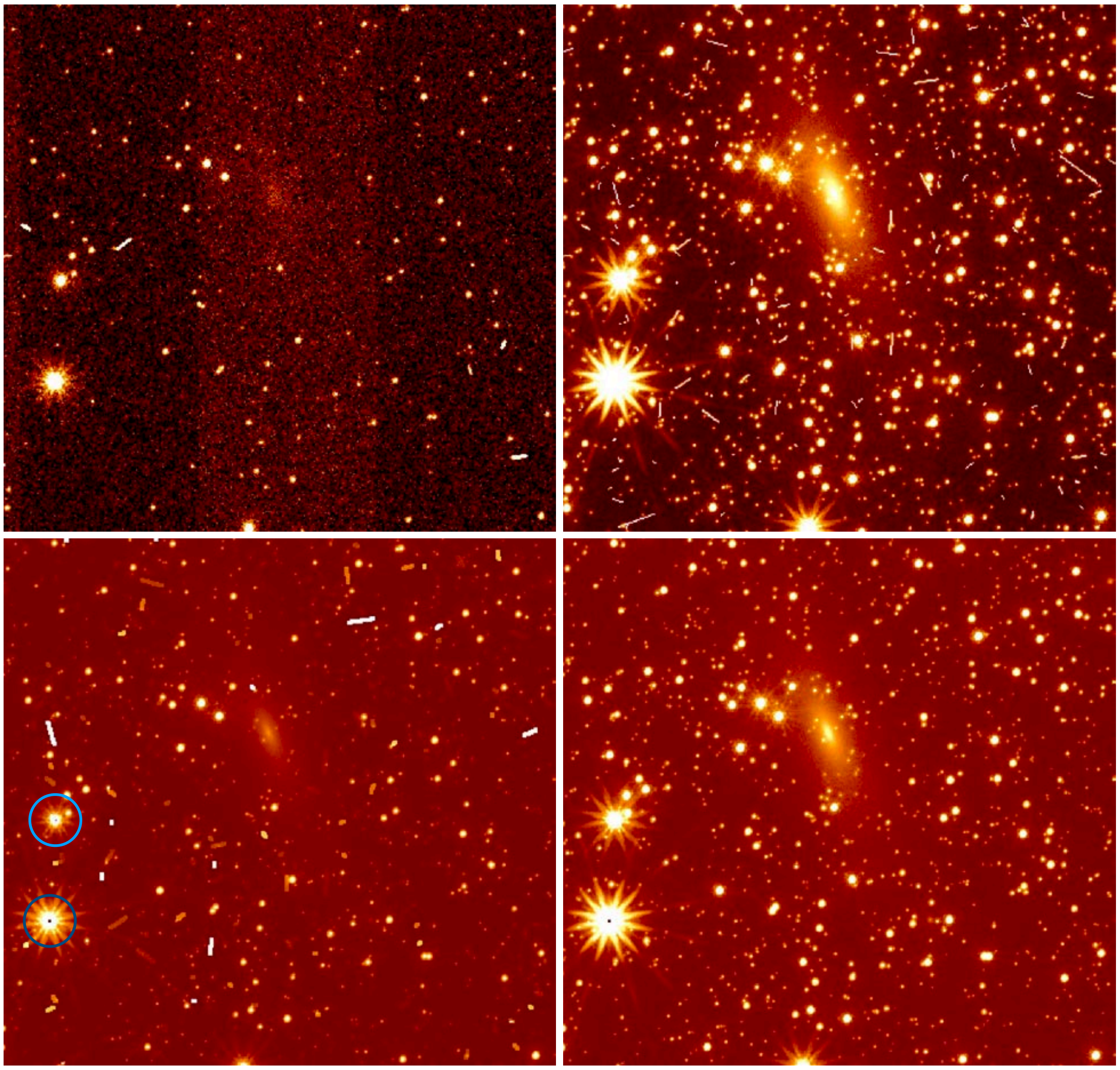


Figure 11: A 400×400-pixel region of SCA 04 is shown for the first and last resultants of the L1 image (top-left and top-right panels, respectively), for the total variance image (extension `_varnce`) on the bottom left and for the L2 calibrated image in the bottom-right panel. Bands or brighter/fainter pixels can be easily seen in the top-left panel and, to a lesser extent, in the top-right panel. This effect is due to slightly different gain values for different read-out amplifiers. Pixels with higher signals or affected by CRs have higher variance (see Eq. 10). Two circles highlight the center of two stars that saturate in different reads. The core of the light-blue star saturates during read 4, leaving only two unsaturated resultants. Following Case (4) in Sect. 3.3, the corresponding variance pixels are flagged to −600 and the L2 pixels are roughly approximated without using the ramp-fitting procedure. The central pixels of the star marked by the dark-blue circle saturate during the second read (first resultant); as a result, the corresponding variance pixels are flagged at −900 and the L2 pixels are set to `LOFLAG`, as described in Case (1).

3.4 Other output files

The program `WFIsim` collects relevant input parameters and save them into an ASCII log file (extension: ".log"), together with additional information on the simulated run, such as the number of added sources to each SCA for both stars and galaxies, jitter parameters, the total effective exposure time, and the effective exposure time of each resultant. In addition, when `SAVE_INPUT_POSITIONS`=Y, two ASCII files per SCA are also saved. They contain the raw SCA positions of added sources, the raw positions in the meta frame, and input instrumental magnitudes for stars, or Sérsic parameters for galaxies. These output source files can be loaded as XY-type region files in SAOImage DS9 (Jole & Mandel 2003) and exactly match sources in the `_LL_SCAXX.fits` scene images. A 4-pixel offset to both X and Y coordinates must be applied to source positions to exactly match L1-based FITS files. The last two columns of the output source files can be used to create DS9 region files for the `_meta.fits` full-frame scene.

4 Performance, caveats and future work

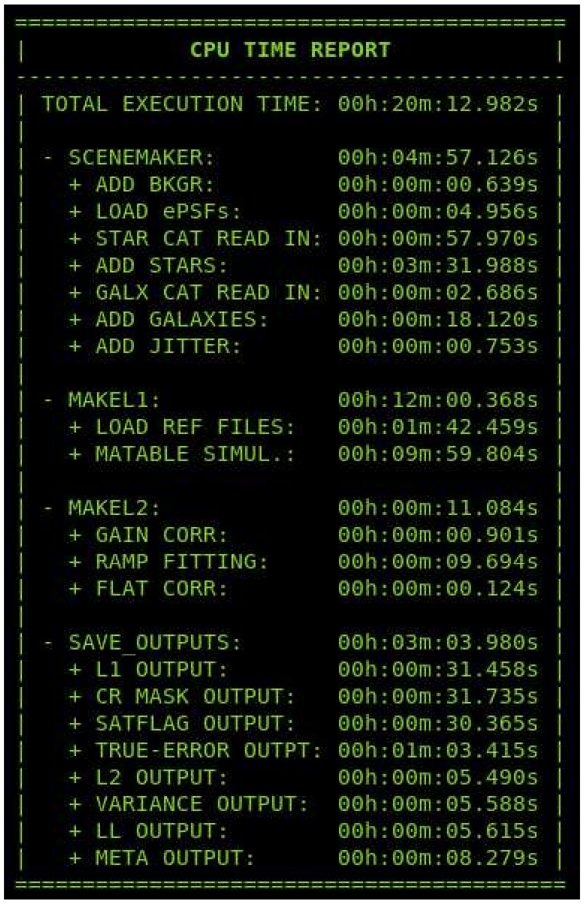
`WFIsim` is currently in alpha stage. As we know more about the Roman mission and the WFI, we will make adjustments to the program and improve its functionality.

`WFIsim`is currently parallelized over the user-defined number of SCAs and, if we exclude I/O read/write time, it takes about the same amount of time to simulate one or 18 SCAs. `WFIsim` has not been fully optimized for performance yet, nor it is particularly resource friendly. We reserve to address any performance and optimization issues after all the simulation steps we plan to include are fully implemented and the code is in a more stable configuration in terms of input parameter definitions.

Because of the way the `makel1` subroutine is set up, `WFIsim` can be memory intense when multiple SCAs are simulated and the `MATABLE` contains many reads. For instance, in the example discussed in Sect. 3 with 18 SCAs and 68 reads, `WFIsim` needs up to 103 GB of RAM. A simple way to reduce memory usage is to simulate fewer SCAs at a time (at the cost of longer total execution time for a full-frame simulation).

On a virtual Linux machine with 18 allocated threads from an Intel Xeon Gold 6154 running at 3.0 GHz and with remotely-mounted hard-disk drives, the simulation described in Section 3 took 20.2 minutes to complete, with most of the time (12 minutes) spent constructing the L1 reads. The total execution time can be cut by more than 50% using modern CPUs (e.g., Intel i9-12900K or AMD Ryzen 9 models) and faster storage devices (e.g., locally-mounted M.2 solid-state drives).

At the end of execution, `WFIsim` prints on screen a time table listing all the main steps of the simulation (when `VERBOSE`=2). The time table for the run described in Sect. 3 is shown



```
=========================================
|             CPU TIME REPORT           |
-----------------------------------------
| TOTAL EXECUTION TIME: 00h:20m:12.982s |
|                                       |
| - SCENEMAKER:         00h:04m:57.126s |
|   + ADD BKGR:         00h:00m:00.639s |
|   + LOAD ePSFs:       00h:00m:04.956s |
|   + STAR CAT READ IN: 00h:00m:57.970s |
|   + ADD STARS:        00h:03m:31.988s |
|   + GALX CAT READ IN: 00h:00m:02.686s |
|   + ADD GALAXIES:     00h:00m:18.120s |
|   + ADD JITTER:       00h:00m:00.753s |
|                                       |
| - MAKEL1:             00h:12m:00.368s |
|   + LOAD REF FILES:   00h:01m:42.459s |
|   + MATABLE SIMUL.:   00h:09m:59.804s |
|                                       |
| - MAKEL2:             00h:00m:11.084s |
|   + GAIN CORR:        00h:00m:00.901s |
|   + RAMP FITTING:     00h:00m:09.694s |
|   + FLAT CORR:        00h:00m:00.124s |
|                                       |
| - SAVE_OUTPUTS:       00h:03m:03.980s |
|   + L1 OUTPUT:        00h:00m:31.458s |
|   + CR MASK OUTPUT:   00h:00m:31.735s |
|   + SATFLAG OUTPUT:   00h:00m:30.365s |
|   + TRUE-ERROR OUTPT: 00h:01m:03.415s |
|   + L2 OUTPUT:        00h:00m:05.490s |
|   + VARIANCE OUTPUT:  00h:00m:05.588s |
|   + LL OUTPUT:        00h:00m:05.615s |
|   + META OUTPUT:      00h:00m:08.279s |
=========================================
```

Figure 12: Example of time table that `WFIsim` prints on screen at the end of execution when `VERBOSE`=2. The table lists the main steps of the simulation and the time spent by the program on them.

in Fig. 12.

In the following, we list current limitations and plans for future improvements.

Caveats:

- The current geometric-distortion model used by `WFIsim` comes from cycle-6 definitions and is now outdated. Raw positions computed by `WFIsim` do not match those obtained with the latest version of the SIAF. The distortion model will be updated in a future version of `WFIsim`.
- The `ROLL_ANGLE` parameter is defined around the nominal aperture of the WFI, which is convenient from an observer point of view, but note that the actual telescope roll angle is defined around its optical axis.
- The way `WFIsim` computes the true errors associated to L1[11] data has not been validated yet. As a result, the output FITS files with extension `_error_SCAXX.fits`,

[11]Note: these true errors are *not* the variance of the L2 data, which is computed at the L2 stage.

which can be obtained by setting `SAVE_TRUE_ERROR`=Y should be treated *cum grano salis*.

- Only the total variance of the ramp-fitting process is currently saved into FITS files. We plan to allow the user to save both read-noise and signal variances in future versions of the code.
- The output FITS files do not contain any type of World Coordinate System (WCS) information. As of now, the only way to obtain sky positions for the image pixels is through the manual cross-identification of sources in the input and output catalogs.
- The headers of all output FITS files only contain bare-minimum information. We plan to expand it and add comment lines relative to the parameters used in the simulation.
- Since the angle of incidence of the incoming light changes over the field of view of the WFI, the effective bandpass shifts in wavelength. This might lead to systematic differences in the measured photometry. Although this effect can be corrected, e.g., using color terms, it is not current implemented in `WFIsim`.
- The maximum radius of the ePSF convolution kernel for galaxies is 45 pixels, since it only concerns with the standard ePSF models. We plan to include wider ePSF models in future releases of `WFIsim`.
- The current implementation of the jitter kernel does not allow for jitter in roll, but only along 2 perpendicular axes at random directions with respect to the X axis of the meta frame. The possibility of having jitter in roll is currently under consideration for future versions of `WFIsim`.

Planned improvements for future releases:

- Allow the user to specify a custom set of SCAs to be simulated, rather than just the upper limit of a range of SCAs that always starts from SCA 01, as it is currently implemented.
- `scenemaker` allows the inclusion of galaxies from outside an SCA if their fluxes within the SCA are still relevant. We plan to do the same for stars.
- Update the geometric-distortion model and improve compatibility with the SIAF files and the python wrapper.
- Include simulated 1/f noise and reference-pixel correction.
- Update the CR generation subroutine using results from JWST sky data.
- Improve the CR mask by allowing the recording all CR events for a pixel.
- Add sky positions to output catalogs when `POINTING_MODE`=RD.

- Combine the four parameters `MXREF/MYREF` and `MRREF/MDREF` into just two whose units are automatically assigned according to `POINTING_MODE`, in a similar way as for `XSHIFT/YSHIFT`.
- Add guide-window functionality. This will include: (i) the ability to define up to 18 input sources as guide stars (one per SCA); (2) automatic placement of guide-windows within the SCAs; (3) guide-window reads; (4) ability to save guide-window data as FITS files; and (5) account for guide-window potential noise increase in L1 data.
- Include persistence effects.
- Expand the pixel-response function definition to account for more effects, such as charge diffusion and brighter-fatter effects.
- Have ePSF models that also depend on the color of sources.

5 Conclusions

The Roman Telescope Branch at the Space Telescope Science Institute is developing a Wide-Field-Imager simulator, `WFIsim`, with the primary goal of scientifically validating Roman calibration pipeline. The simulator is designed to reproduce realistic level-1 and level-2 data products. This technical report contains a detailed description of all the input and output files, and provides a step-by-step explanation of the main subroutines, from the generation of the astronomical scene to the ramp-fitting process. As we learn more about the Roman mission and its wide-field detector, we will expand the capabilities of the simulator and make the necessary changes to ensure high levels of realism in the output data.

While the main goal of `WFIsim` is to work together with the calibration pipeline, the software is mature enough that can now be useful for detector-characterization studies and calibration purposes.

Bibliography

Anderson, J. & King, I. R. 2000, PASP, 112, 1360

Anderson, J. & King, I. R. 2006, “PSFs, Photometry, and Astrometry for the ACS/WFC”, Instrument Science Report ACS 2006-01 (Baltimore: STScI)

Bellini, A. 2018, “Will Gaia be precise enough to solve for the geometric distortion of the WFI?”, Technical Report WFIRST-STScI-TR1801 (Baltimore: STScI)

Casertano, S. 2022, “Determining the best-fitting slope and its uncertainty for up-the-ramp sampled images with unevenly distributed resultants”, Technical Report Roman-STScI-000394 (Baltimore: STScI)

Ciotti, L. 1991, A&A, 249, 99

Greenfield, P., Droettboom, M., & Bray, E. 2015, A&C, 12, 240

Giavalisco, M., Ferguson, H. C., Koekemoer, A. M., et al. 2004, ApJ Letters, 600, 93

Jole, W. A. & Mandel, E. 2003, Astronomical Data Analysis Software and Systems XII, 295, 489

Miles, N. D., Deustua, S. E., Tancredi, G., et al. 2021, ApJ, 918, 86

Peng, C. Y., Ho, L. C., Impey, C. D., et al. 2002, AJ, 124, 266

Robberto, M. 2010, “A library of Simulated Cosmic Ray Events Impacting JWST HgCdTe Detectors”, Technical Report JWST-STScI-001928,SM-12 (Baltimore: STScI)

Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363

Appendix A. Content of the input configuration file used in Section 3

```
MODE=L2
SCA_NUMBER=18
POINTING_MODE=RD
MRREF=79.965
MDREF=30.031
ADD_DITHER_NOISE=Y
DITHER_NOISE_VALUE=0.01
ROLL_ANGLE=8.0
ADD_JITTER=Y
ADD_SCENE=Y
ADD_STARS=Y
STARCAT_NAME=/user/username/stars.cat
NSTARS=-1
ADD_GALAXIES=Y
GALCAT_NAME=/user/username/galaxies.cat
NGALS=-1
FAINT_SIGMA_THRESHOLD=1.1
ADD_BACKGROUND=Y
BACKGROUND=1.5
FILTER=F158
PSFPATH=/user/username/epsfs/
MATABLE=ITTRTRTRRRTRRRTRRRRRRRRTRRRRRRRRTSSSRRRRRRRRTSSSRRRRRRRRTSSSRRRRRRRRT
ADD_CR=Y
USEFLAT=Y
FLATPATH=/user/username/flat/
USEDARK=Y
DARKPATH=/user/username/dark/
USEBIAS=Y
BIASPATH=/user/username/bias/
USEGAIN=Y
GAINPATH=/user/username/gain/
USECNL=Y
CNLPATH=/user/username/cnl/
USEDQ=Y
DQPATH=/user/username/dq/
USESAT=Y
SATPATH=/user/username/sat/
USERN=Y
RNPATH=/user/username/rn/
OUTPUT_ROOTNAME=test
SAVE_INPUT_POSITIONS=Y
SAVE_JITTER_KERNEL=Y
SAVE_LL=Y
SAVE_L1=Y
SAVE_CRMASK=Y
SAVE_TRUE_ERROR=Y
SAVE_SATFLAG=Y
SAVE_L2=Y
SAVE_VARIANCE=Y
SAVE_META=Y
```