STScI | SPACE TELESCOPE SCIENCE INSTITUTE

Instrument Science Report WFC3 2021-11

A maximum likelihood approach to estimating the flux in infrared detectors non-destructive ramps.

M. Gennaro, H. Khandrika

July 14, 2021

Abstract

Some types of infrared detectors record the value of accumulating charge while being actively exposed to light. The charge is measured, or sampled at various time intervals providing the user with what is usually referred to as a ramp. The incident flux can be derived as the slope of the ramp, i.e., the slope of the charge vs. time line, an approach called up-the-ramp fitting. The `CRCORR` *task in the* ***calwf3*** *pipeline performs the up-the-ramp fitting for the HST WFC3/IR instrument. During the fit process, the pipeline identifies and flags CR hits, seen as discontinuities in the otherwise linear ramps. In an attempt to fully characterize the pipeline performance, we derive a likelihood-based approach to the signal determination problem. Our approach is general and can be applied to JWST detectors read sequences as well. Writing the full likelihood allows for a proper accounting of the different noise sources and their covariance: the Poisson noise for the signal, the Gaussian read noise, and the uniform quantization and bit-shift noise introduced by the digitization and (for JWST ramps) on-chip multi-frame averaging. Using simulated WFC3/IR ramps, we verified that both our method and* ***calwf3*** *match the theoretical noise expectations.* ***calwf3*** *is however under-efficient in detecting cosmic-rays. In the high Signal to Noise Ratio (SNR) regime, this leads to a small bias in the* ***calwf3*** *results. In the low SNR regime some differences appear: the* ***calwf3*** *weighted least squares approach is approximately equivalent to maximizing a Gaussian likelihood, leading to the fitted flux being sometimes negative, a non-physical, absurd result. Our treatment of the signal as a Poisson variable avoids the non-physical negative results. Both* ***calwf3*** *and our method implicitly assume that the signal is constant during each exposure. We test what happens when such an assumption is broken by simulating a time-variable signal. We find that both methods are good at determining the time-averaged flux, but* ***calwf3*** *has no way to determine whether the flux is truly constant during an exposure. In our*

*approach, we introduce a goodness-of-fit criterion, that can be adapted and implemented in **calwf3** as a simple χ^2 test, to efficiently flag ramps with variable signals. Finally, we compare our results to predictions from weighted least squares as well as generalized least squares methods. The first is the default method adopted in the JWST pipeline, while the second is available for users who wish to reprocess their data. We concur with previous findings that the weighted (even the so-called optimally weighted) least squares method underestimates the uncertainties, due to its neglecting the correlated noise between the samples. We find very good agreement between our estimates and the results of the generalized least-squares approach.*

---

1 Introduction

Several existing types of infrared (IR) detectors exploit semiconductor technology that allows measuring the accumulated photo-electrons while an exposure is still ongoing. This approach is called non-destructive reading, or sampling up-the-ramp. The ramp here is the run of the accumulated charge over time. In many astronomical cases, the incident flux on a pixel is constant during an exposure. Therefore the accumulated charge in a pixel increases linearly with time, and a plot of the total charge accumulated versus time looks like a straight line, with the intercept being the bias level, and the slope being the flux in counts (or ADU, or DN) per second. Note that even when exposed to a constant flux, IR detectors usually manifest a non-linear behavior as the accumulated charge approaches the pixel full well capacity. This type of non-linearity can be partially calibrated and corrected for, and in the rest of this paper we implicitly assume that constant flux equals linear (or linearized) ramps.

It would be very tempting to belittle the problem of determining the incident flux to a simple fit-a-straight-line problem. On the contrary, the accurate treatment of the noise as well as the mathematical foundation of the fitting algorithm are very important details, and have spun a vast literature on the topic.

In this work, we treat the problem exactly, using a full likelihood maximization approach. Our aim is to validate the up-the-ramp fitting results of the **calwf3** pipeline, in particular the `CRCORR` step of the `wfcir` branch of **calwf3**. The `CRCORR` step, as the name suggests, corrects for Cosmic Ray (CR) events. When a CR hits a pixel in an IR detector such as the one used by the WFC3/IR camera, it creates an instantaneous jump, or discontinuity in the accumulated signal versus time. Identifying such a jump allows splitting the ramps in two parts (or three, four, etc. as the number of identified CRs increases), and work with each part of the ramp separately. The underlying idea is that each CR hit perturbs only the charge value accumulated between the frame preceding the hit and the frame immediately following it. After the hit, the signal will again accumulate in a linear fashion. A CR hit results in a loss of signal-to-noise ratio (SNR) related to the length of the affected interval relative to the total integration time: the values of the charge at the beginning and end of the affected interval still provide useful information, however the interval in between is effectively lost. Slopes can be determined for the separate parts of the ramp, including the extrema of the CR-affected intervals. Individual slopes can then be averaged together. This is in fact one of the most prominent operational advantages of IR detectors over CCDs: a

CR hit does not necessarily mean a total loss of information, but just a partial loss of a single interval. After identifying CRs, the `CRCORR` step also determines the slope value, and performs a weighted average of the slopes.

The `CRCORR` step of **calwf3** is directly inherited from the corresponding step in **calnica**, the pipeline for processing *HST*/NICMOS data. A description of the **calnica** implementation of up-the-ramp fitting can be found in Dahlen et al. (2008). In that document, a detailed implementation of optimum weights for straight line fit in different SNR regimes is introduced. These optimum weights follow the prescription by Regan (2007). The latter are a numerical approximation to the weights derived by Fixsen et al. (2000).

In an exhaustive series of papers, Dr. M. Robberto approaches the up-the-ramp fitting problem as a generalized least squares problem, where the unknowns are the slope and intercept of the accumulating signal line (Robberto, 2009a,b,c, 2014); the last of these publications can be considered the compendium of all the preceding ones, and for ease of reading, we will refer only to it in the following, implicitly remembering the other papers in the series. For completeness, we mention that a similar approach to up-the-ramp fitting can be found in Rauscher et al. (2007).

While Fixsen et al. (2000) dealt with the difference between consecutive frames, Robberto (2014) uses the total accumulated signal as the dependent variable. The two approaches are equivalent, even though it is important to recognize the differences:

- In Fixsen et al. (2000), the starting point is that the signal accumulated between two consecutive frames is independent from the signal accumulated between any other two consecutive frames, including back-to-back pairs of frames. Such signal is estimated by using the difference between two consecutive frames. Although the number of accumulated photo-electrons between consecutive frames is truly an independent realizations of a Poisson process, one must remember that the measurement process introduces a correlation (via the read noise) between successive differences. Thus the Fixsen et al. (2000) correlation matrix, which describes the noise for the differences between consecutive frames, has non-zero first upper and lower minor diagonals with all elements equal to $-\sigma_{ron}^2$, where σ_{ron} represents the standard deviation of the read-noise per frame.

- Robberto (2014) uses the total integrated signal as dependent variable. In that case, there is no read-noise correlation between consecutive data points (i.e. each read process is independent from all the others), however the signal itself is correlated. If y_i, y_{i+1} are the number of accumulated photons up to times t_i, t_{i+1}, with $t_i < t_{i+1}$, then $\mathrm{Cov}(y_i, y_{i+1}) = y_i$.

The Robberto (2014) approach also deals with an added complexity, introduced by using modern near-IR detectors electronics to average multiple frames into a single group on-chip."Multiple averaged frames per group" is a typical operating mode for the *JWST* detectors. Consecutive frames may be averaged together, and some frames may be skipped altogether, providing two advantages: first the readnoise is effectively reduced, second the amount of data produced is smaller, which is an advantage for storage and for downlink as

well, in the case of space-based missions. Frame averaging adds further terms to the covariance matrix. Robberto (2014) derives the complete covariance matrix for this operating mode, adding the quantization and bit-shift noise associated to ADC and on-chip average respectively.

Our approach is similar to that of Fixsen et al. (2000) in that we also treat the accumulated photo-electrons between two frames as the dependent variable. However, although we focus mostly on validating the results from the **calwf3** pipeline, we extend our method to the case of multiple frames per group. We thus follow a similar path to Robberto (2014) in deriving the full variance-covariance matrix in the presence of multiple averaged frames. Our fitting strategy is however completely different from Fixsen et al. (2000), Dahlen et al. (2008) and thus **calwf3**, as well as from Robberto (2014) .

It is important to note that the generalized least squares approach by Robberto (2014) and the optimum-weights approach by Fixsen et al. (2000), Regan (2007), Dahlen et al. (2008), and consequently the one used in **calwf3** are substantially equivalent when applied to the case of WFC3/IR read sequences (i.e., no skipped nor averaged frames). The underlying similarity lies in that all approaches are in fact assuming an underlying Gaussian noise model, where the variance of the signal and the read noise are added in quadrature to give the total variance-covariance matrix, which enters the likelihood as (part of) the Gaussian exponent. Although these approaches do not explicitly deal with likelihoods, they solve the flux determination problem within the least-squares minimization framework. This is equivalent to maximizing a Gaussian likelihood, because minimizing the sum of the squared residuals is equivalent to maximizing the log-likelihood with respect to the unknown mean for a normal distribution.

In practice, this approach approximates the Poisson likelihood of a signal with mean λ, with a Gaussian of the same mean and $\sigma = \sqrt{\lambda}$. The shortcoming of this approximation is that it fails for $\lambda \to 0$, i.e. for small signals. Our approach is fundamentally different in that it explicitly expresses the likelihood as a product of a Poisson distribution for the signal and a Gaussian distribution for the readnoise, as well as a uniform distribution for the quantization and bit-shift noises. In this respect, our approach is an exact description of the process under consideration, and should thus be preferable. However, there are practical considerations (mostly computational speed) that fully justify the fact that, in practice, the Gaussian approximation is a good working approximation, as we will see below.

This paper is organized as follows: Section 2 presents the theory for deriving a full likelihood expression for a ramp measurement process. The theory is expanded in Section 3 to the case of the on-chip average of multiple frames. Section 4 presents our iterative solution for maximizing the likelihood, followed by Section 5 which describes both the **calwf3** and our own approach to the problem of flagging CR events. Section 6 illustrates the detailed calculations for the variance-covariance matrix used in our algorithm and Section 7 goes into the details of our algorithm implementation. Section 8 shows how to use our code to simulate IR ramps, and Section 9 shows how to use it to perform a fit and obtain an estimate of the flux. Section 10 compares the performance of our approach with the **calwf3** pipeline. Section 11 shows a similar comparison with a generalized least squares approach similar to that available (but not by default) for the *JWST* pipeline. We summarize our findings and conclusions in Section 12.

The Appendix provides additional examples for using our code. The latter is publicly

available at `https://github.com/spacetelescope/Up_the_ramp`. The repository contains both the main code as well as some example notebooks to showcase its use.

2 The likelihood of an IR ramp measurement process

Let f be the flux in photons per second expected on an individual pixel of an IR detector. f is determined by the source's true flux, $\mathcal{F}$ (measured in physical units, e.g. in erg s^{-1} cm^{-2} Å^{-1}) and by all intervening factors, such as system throughput, optical system collecting area, quantum efficiency and so on, that we assume are all quantified in a single number, the sensitivity, s. Thus if f is measured, $\mathcal{F}$ in known as $\mathcal{F} = fs$. The number of photo-generated electrons in a given time interval Δt is an integer number, Δx, with Poisson distribution with mean $\lambda = f\Delta t$, where we assume that 1 electron-hole pair is generated per photon (the detector's quantum efficiency is incorporated in the sensitivity, s). The distribution of photo-electrons is thus:

$$P(\Delta x) \sim Poi(\lambda) = \frac{e^{-\lambda}\lambda^{\Delta x}}{\Delta x!} \tag{1}$$

We keep the Δ notation for convenience, to explicitly indicate that this is the charge accumulated between two instants straddling the time interval Δt. The two instants may correspond to two read samples up-the-ramp. Each sample, when extended across all pixels of a detector is also referred to as a frame.

When measuring the accumulated charge, the detector electronics add a Gaussian random read-noise with standard deviation σ_{ron}, usually expressed in electrons. The readout operations at the beginning and at the end of the interval Δt are independent. Thus if the true accumulated charge is $\Delta x = x_e - x_s$, where x_e and x_s indicate the values end and start of the time interval $\Delta t = t_e - t_s$, and the readout values are y_e and y_s, we have that the likelihood of measuring $\{y_e, y_s\}$ is:

$$P(y_e, y_s|\lambda) = \sum_{x_s}\sum_{x_e} P(\Delta x|\lambda) \times P(y_e|x_e) \times P(y_s|x_s) \tag{2}$$

where $P(\Delta x|\lambda)$ is given by equation (1) and $P(y|x)$ is the distribution of measured counts given the real ones. The latter is a normal distribution with $x - y \sim \mathcal{N}(0, r^2)$, i.e. the difference between measured and real counts is distributed like a normal distribution with mean 0 and variance σ^2_{ron}. The double sum extends over all acceptable values of x_s, x_e. The latter are unknown, or latent variables, thus all their possible values must be accounted for in order to obtain the total probability of the measured y's. The x_i's can only assume integer values. The starting accumulated charge x_s runs from 0 (or from a known bias level) to the pixel full-well capacity value. The ending accumulated charge, x_e runs from x_s to the pixel full well capacity, i.e. the accumulated charge cannot decrease with time. In words, the probability of measuring $\Delta y = y_e - y_s$ counts in the interval of time Δt is given by the product of the Poisson probability of generating $\Delta x = x_e - x_s$ electrons given a true flux, f, and $\lambda = f\Delta t$, times the Gaussian probability of reading out the numbers y_e, y_s, if the true numbers are x_e, x_s and each read operation has associated readnoise variance σ^2_{ron}. All of this is summed over all acceptable values of x_e and x_s.

When N samples are recorded up-the-ramp we have:

$$P(y_1, y_2...y_N|\lambda) = \sum_{x_1} ... \sum_{x_N} P(\Delta x_{1,2}|\lambda) \times ... \times P(\Delta x_{N-1,N}|\lambda) \times \\ P(y_1|x_1) \times ... \times P(y_N|x_N); \tag{3}$$

where we explicitly set $\Delta x_{i,i+1} = x_{i+1} - x_i$ for clarity. In this case, the summation limits are $0 \leq x_1 \leq$ full well, $x_1 \leq x_2 <$ full well, and so on. Thus for N samples, the underlying random events are $N-1$ realizations of a Poisson process with mean λ, and N realizations of a Gaussian process with mean 0 and standard deviation σ_{ron}.

A further complication is introduced by the fact that sampling up-the-ramp does not necessarily happen at regularly spaced times. The WFC3/IR detector operations indeed allow the user to select sampling sequences from a wide range of possibilities, some including irregular timing. The plan for *JWST* operations is that all *JWST* detectors are instead going to always be sampled at regular cadence. Equation 3 can be generalized to an unevenly sampled case, by explicitly taking into account the values $\Delta t_{i,i+1} = t_{i+1} - t_i$. The number of photo-electrons in each interval is still a realization of a Poisson variable, but we have different Poisson distributions (different means) within each interval, i.e. $\lambda_{i,i+1} = f\Delta t_{i,i+1} \neq \lambda_{j,j+1} = f\Delta t_{j,j+1}$ if $\Delta t_{i,i+1} \neq \Delta t_{j,j+1}$. In this case equation (3) becomes:

$$P(y_1, y_2...y_N|f) = \sum_{x_1} ... \sum_{x_N} P(\Delta x_{1,2}|f\Delta t_{1,2}) \times ... \times P(\Delta x_{N-1,N}|f\Delta t_{N-1,N}) \times \\ P(y_1|x_1) \times ... \times P(y_N|x_N); \tag{4}$$

where we made the dependence on the flux, f, more explicit. We remind the reader that:

$$P(\Delta x_{i,j}|f\Delta t_{i,j}) \quad \Longrightarrow \quad \Delta x_{i,j} \sim Poi(f\Delta t_{i,j}) \tag{5}$$

$$P(y_i|x_i) \quad \Longrightarrow \quad x_i - y_i \sim \mathcal{N}(0, \sigma^2_{ron}) \tag{6}$$

Equation (4) can be further rewritten. The sums on the right hand side correspond to marginalizing the joint probability of the $y_1, ..., y_N$ and the $x_1, ...x_N$ over the $x_1, ...x_N$, i.e.:

$$P(y_1, y_2...y_N|f) = \sum_{x_1} ... \sum_{x_N} P(y_1, y_2...y_N, x_1, ...x_N|f). \tag{7}$$

This expression allows us to clearly see that the x_i's, i.e. the true values of the accumulated charge, can be considered as nuisance parameters. Such values are not observed, but they clearly enter the problem. In principle, the likelihood in equations (4) or (7) can be explicitly computed, given a value of f. The t_i's are known from the timing sequence of the detector, and the y_i's are measured. The probability distribution functions on the right hand side of the equations are known (normal and Poisson distributions) and can be explicitly evaluated. However, the practical problem is that the sums run from 0 to the pixel full well (about 80,000 electrons in WFC3/IR) for each of the x_i's. The computational problem rapidly explodes even for a moderate number of samples.

Instead of working with the marginalized probability, i.e. the left hand side of equation (7), we proceed to derive the true flux, f, by maximizing the full probability, including

the latent variables, i.e. $P(y_1, y_2...y_N, x_1, ...x_N|f)$. More correctly, using Bayes' theorem, the joint likelihood of the y_i's and x_i's can be "inverted" to give the posterior probability of the flux:

$$P(f|y_1, y_2...y_N, x_1, ...x_N) \propto P(y_1, y_2...y_N, x_1, ...x_N|f) \times P(f) \tag{8}$$

Any prior knowledge on the flux can be thus used to derive the posterior. In the following we utilize a uniform prior on the flux, effectively reducing the problem of finding the maximum a posteriori flux, to a likelihood maximization problem. We also explicitly write the first factor in eq. (8) as:

$$\begin{aligned} P(y_1, y_2...y_N, x_1, ...x_N|f) &= P(\vec{y}, \vec{x}|f) \\ &= P(\vec{y}|\vec{x}) \times P(\vec{x}|f) \\ &= \prod_{i=1}^{N} \mathcal{N}(x_i - y_i; 0, \sigma^2_{ron}) \times \prod_{i=1}^{N-1} Poi(\Delta x_{i,i+1}; f\Delta t_{i,i+1}) \end{aligned} \tag{9}$$

.

Where:

- in the first step we have made use of the fact that only the true number of accumulated electrons within an interval depends *explicitly* on the flux
- we have used equations (5) and (6) to display the probability distributions
- we have made it explicit that the product of the normal distributions runs over all the N frames
- we have made it explicit that the product of the Poisson distributions runs over all the $N-1$ intervals
- we are using all intervals and frames, neglecting possible hits by CRs (but see Sec. 5)

We will modify equation (9) in Section 3 to account for averaged frames.

3 Generalizing to multiple frames averaged groups (the JWST case)

The detectors on *JWST* will be continuously read out at a constant rate. E.g. all 10 of the 2048×2048 HgCdTe detectors of the NIRCam instrument will be read every ~ 10.8 seconds when operated in full-frame imaging mode. For long integrations a very large amount of data would be created if $10 \times 2048 \times 2048$ 16-bit digital numbers were to be saved every 10 seconds. This amount of data would quickly saturate the solid state recorders on-board *JWST*. A solution is to save only a fraction of the frames, and to skip saving (but not reading) some, while the charge keeps accumulating on the pixels.

The NIRCam electronics allow for on-chip average of a given number of consecutive frames. Skipped frames can be interleaved with the averaged frames. A series of m averaged frames and s skipped frames is called a group. A single value per pixel is saved for each

group. This value corresponds to the average value of the charge values measured in the m averaged frames, each with its read noise.

The on-chip average happens through a sum-and-bit-shift operation. The readout values of m frames are first summed together, and then the binary numbers in the detector electronics memory registers are shifted by $n_{shifts} = \log_2 m$ bits. This operation is equivalent to integer-dividing the sum by m. Other than saving data volume, the average operation reduces the readnoise variance in the averaged frame by a factor of $1/m$ (the read noise is uncorrelated between frames). At the same time the averaging operation changes the expression for the noise of the single group value due to:

- correlated noise between the successive m frames, which must be taken into account when propagating uncertainties on the averaged values and on the differences between consecutive averaged values.
- uncorrelated read noise between the m frames within a group, which leads to an effective read noise per group with variance of σ^2_{ron}/m
- a rounding error due to the fact that the division by m is actually performed as an integer division

The first of these bullet items will become important in Section 7.3 and will be ignored here. The other two bullets will find instead immediate application in the current Section.

The truly independent, Poisson variables in a sequence of averaged groups are the numbers of electron accumulated between subsequent individual frames. For this reason equation (9) needs to be modified:

$$P(\vec{y}, \vec{x}|f) = P(y_1, y_2, ...y_N, x_{1,1}, x_{1,2}, ...x_{1,m}, x_{2,1}, x_{2,2}, ...x_{2,m}...x_{N,1}, x_{N,2}, ...x_{N,m}|f) \quad (10)$$

where as in equation (9) the measured quantitites are $\vec{y} = y_1, y_2, ...y_N$. However here we made it explicit that the latent variables $\vec{x}$ are $N \times m$ where m is the number of frames averaged together in each group and N is the total number of groups. The explicit expression for this probability can be obtained by defining the average electrons per group:

$$z_i = \lfloor \frac{\sum_{k=1}^{m} x_{i,k}}{m} \rfloor \quad (11)$$

The $\lfloor . \rfloor$ operator indicates the *floor* function. As mentioned above the on-chip average operation is equivalent to an integer division, which in turn can be obtained by using a regular division and then discarding the decimal remainder using the floor operation. Using the z variables, equation (9) becomes:

$$\begin{aligned} P(\vec{y}, \vec{x}|f) &= P(\vec{y}|\vec{x}) \times P(\vec{x}|f) \\ &= P(\vec{y}|\vec{z}(\vec{x})) \times P(\vec{x}|f) \\ &= \prod_{i=1}^{N} \mathcal{N}(y_i - z_i; 0, \frac{\sigma^2_{ron}}{m}) \times \prod_{i=1}^{m \times N - 1} Poi(\Delta x_{i,i+1}; f * \Delta t_{i,i+1}) \end{aligned} \quad (12)$$

Here we explicitly indicated that the averaged $\vec{z}$ depend on the individual frames $\vec{x}$, however the $\vec{y}$ values do not depend on the values of $\vec{x}$ individually, but only on their grouped average. Thus two sets of $\vec{x}$, although different, could potentially produce the same $\vec{z}$. In some cases, different $\vec{x}$ could produce even the same total probability, if, e.g. we have permutations of the $x_{i,m}$ values within the same i group, such that also the Poisson probabilities product stays the same. In equation (12) we also indicated that the normal distribution has a variance set by the effective readnoise, equal to the actual readnoise variance divided by the number of average frames, m.

It must be noted that the product of the Poisson distributions runs over all individual consecutive frames pairs, not just those within the same group. Thus the $(i, i+1)$ pair will sometimes involve two frames within the same group and sometimes the last frame from a group and the first from the next. In the former case the $\Delta t_{i,i+1}$ interval will be the typical sampling interval (e.g. 10.8 seconds for NIRCam full frame), while in the latter case it will be $(s+1)$ times as much, where s is the number of skipped frames.

Incidentally, due to the floor operation of equation (11), two sets of $\vec{x}^{(1)} \neq \vec{x}^{(2)}$ values in which the sum of one (or more) group values differ by less than m, could have the same likelihood if $z_i^{(1)} = z_i^{(2)}$. This in practice guarantees that the rounding error associated to the on chip average (third bullet in the list in this section) is implicitly included into our expression of the likelihood.

4 An iterative algorithm for maximizing the likelihood

As mentioned in Section 2, working with the joint probability $P(y_1, y_2...y_N, x_1, ...x_N|f)$ circumvents the large summation problem of equations (4) or (7). However, given that the joint probability depends explicitly on the x_i's, they have to be solved for at the same time as solving for the flux, f. For this, we devise a computation method graphically described in Figure 1 and explained in the remainder of this Section.

Leaving the details to following sections, we illustrate here the core ideas of the algorithm. The method consists in starting from a good guess for the flux value, f^{old}, for example the average of the individual electron rates from each sampling interval. The latter can be obtained by first setting the x_i, i.e the true values of the accumulated charge -without readnoise- to the measured y_i values. We then estimate the flux as the measured electrons difference $x_{i+1} - x_i$ divided by the time differences $t_{i+1} - t_i$. In Figure 1, the header *old* denotes a value at the beginning of an iteration, while the header *new* denotes the new value computed at the end of an iteration.

For each iteration we proceed as follows: we find the values of the true photo-generated electrons $x_1, ..., x_N$ that maximize the likelihood in equation (9), or its equivalent for multiple averaged frames equation (12), at fixed flux $f = f^{old}$. Given the new values of $x_1, ..., x_N$, we compute a new value of the flux, f^{new}. The latter is compared to f^{old}, and if the two are within a fixed threshold, ϵ, the iterative method halts. Otherwise the *new* value of the flux becomes the *old* one and we proceed to the next iteration. In summary, we are iteratively adjusting the non-observed, nuisance parameters -the true number of photo-electrons produced in each interval- and the flux, to maximize the full likelihood of both observed and nuisance parameters as a function of the flux. Section 7 and its subsections

Data: $y_1...y_N$, measured counts
$t_1...t_N$, sampling times
Result: f, true flux
Initialization: $x_i^{old} \leftarrow y_i$
$f^{old} = \langle \frac{x_{i+1}^{old} - x_i^{old}}{t_{i+1} - t_i} \rangle$;
while $j < Maxiter$ **do**
$x^{new} = \arg\max_{x_1...x_N} P(y_1, y_2, ...y_N, x_1, x_2...x_N | f^{old})$;
$f^{new} = \langle \frac{x_{i+1}^{new} - x_i^{new}}{t_{i+1} - t_i} \rangle$;
if $|f^{new} - f^{old}| < \epsilon$ **then**
stop ;
else
$f^{old} = f^{new}$;
$j = j + 1$;
end
end

Figure 1: Simplified schematics of our Likelihood Maximization procedure

further clarify the details of the algorithm implementation.

5 Cosmic rays

As mentioned before, one of the main advantages of IR detectors is their ability to be readout in non-destructive read mode. Under this mode Cosmic ray (CR) hits do not necessarily mean a complete loss of information, as opposed to the case of CCDs. Nevertheless, CRs need to be identified and flagged by any algorithm that aims at measuring the incident photon flux.

5.1 The calwf3 approach to identifying CRs

The `CRCORR` step in the **calwf3** pipeline identifies CRs by computing an initial slope value. It then computes the difference between the expected number of counts at each sample time and the measured counts. A CR hit corresponds to a jump, or discontinuity, on top of the linear run of counts versus time. An initial line fit without flagging CR hits would overestimate the counts before the hit and underestimate those immediately after, as illustrated in Figure 2. The **calwf3** pipeline flags CRs by:

1. computing the residuals between the predicted and the observed fluxes
2. dividing the difference by the estimated error on the difference
3. scanning these normalized residuals starting from the beginning of the exposure

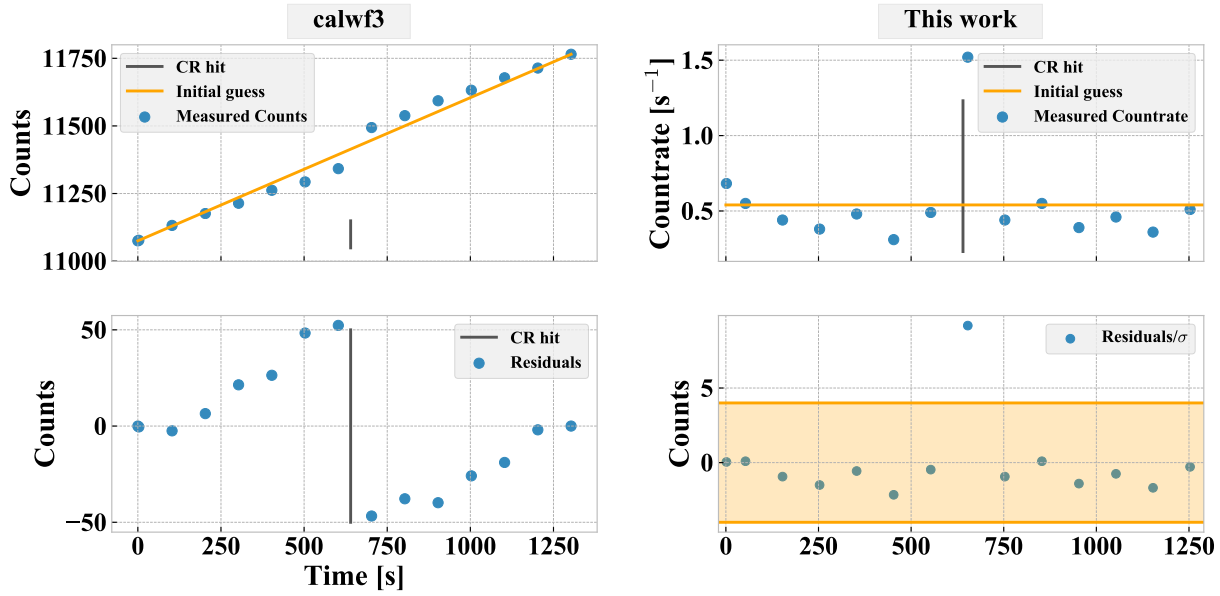

Figure 2: Top, left: simulated ramp (blue circles), with a CR hit happening half-way. The CR deposited counts are indicated by the height of the vertical gray line. Yellow line: initial guess from **calwf3**, before CRs are identified. **Top, right:** the same ramp, displayed as count rate, by taking the difference between subsequent frames and dividing it by the interval times. The count rate is almost constant, with the exception of the interval with the CR hit. The gray line has been normalized by the interval time as well. Yellow: initial estimate in our scheme, computed simply as the average of the count rate estimates from individual intervals. **Bottom, left:** residuals between the initial guess and the observed counts for the **calwf3** case. The residuals abruptly change sign when crossing the CR hit time. Such change in the absolute value of the residuals relative to the expected change due to the estimated noise is used to flag CR hits in **calwf3**. **Bottom, right:** our CR flagging approach. We compute the difference between the measured counts in each interval multiplied by the gain minus the expected number of electrons given the current estimate of the flux. This difference is divided by the estimated noise, and the intervals with normalized residuals outside the 4σ confidence region (yellow shaded area) are flagged for CRs hits.

4. if the absolute difference between two consecutive residuals is larger than a threshold, a CR hit is identified

This means that the residuals must change relative to each other by more than a given threshold. The default threshold used by **calwf3** is 4 (unitless, because the differences are divided by the estimated error), and it is contained in the `CCREJTAB` reference file, in a variable named `CRSIGMAS`. While this approach makes sense, intuitively, it is hard to quantify the outcome of the CR identification procedure in statistical terms. Moreover, as we will see in Section 10 the **calwf3** approach is very underefficient in detecting cosmic rays, causing a large number of false negative detections, which in turn can bias the output value of the flux.

5.2 Our approach to identifying CRs

We have followed a different, more easily quantifiable and statistically justifiable, approach. At each iteration we use the current estimate of the flux to compute the standard deviation of the expected number electrons within each interval. The total standard deviation is equal to the sum in quadrature of three independent sources of noise:

1. The photon noise associated to the true number of electrons between two reads. Its variance is equal to the flux times the time interval for detectors like WFC3/IR, where each recorded sample is the results of a single read operation. In such case the true number of electrons accumulated in a sampling interval follows a Poisson distribution with mean (and variance) equal to the flux times the interval length. For *JWST*-like operations, where multiple frames may be averaged on-board to form a group, the variance contains additional terms related to the covariance between the averaged frames. Although the number of electrons accumulated between reads follows a Poisson distribution, the number of electrons resulting from taking the difference between two averaged groups does not. The full expressions for these variance terms are detailed in Section 6.

2. The readnoise. When taking a difference between frames, in order to measure the accumulated electrons in a single interval, the corresponding noise variance is twice the single-read one, i.e. $2\sigma^2_{ron}$. For *JWST*-like operations where multiple frames are averaged, each with readnoise $\sim \mathcal{N}(0, \sigma^2_{ron})$, the resulting readnoise for the difference of two groups has variance equal to $2\sigma^2_{ron}/m$, where m is the number of averaged frames.

3. Quantization and bit shift noise. When digitizing the voltage at each pixel's amplifier into a DN (digital number), a rounding error is added. The variance of such error is the variance of a uniform distribution on a unit interval, i.e. $\sigma^2_q = 1/12$. The variance must be multiplied by the square of the gain, when estimating the error on the number of electrons. Finally it must also be multiplied by 2, to account for the 2 independent samples that are being subtracted. In *JWST*-like operations, a further multiplication factor is the square of the number of averaged frames. The quantization error affects the single frames, and thus it adds in quadrature when multiple frames are summed together in the on-chip average scheme used on *JWST*.

The explicit derivation of these three terms is detailed in Section 6.2. The three variance terms are summed in quadrature, and we take the square root of the result as an estimate of the standard deviation of the measured number of electrons in each interval. We compare the expected number of electrons to the measured number of counts between two groups multiplied by the gain. We take the difference between the two and divide it by the standard deviation computed above. We then use a tunable threshold, δ to flag outliers; the default value for δ is 4. In this case CRs hits are seen as positive outliers in the estimated count rate over a certain threshold. This type of flagging is more amenable to a tuning based on statistical arguments such as, completeness (false negatives) vs. purity (false positives) in the desired number of flagged events, as we show in Section 10.4. While the threshold itself is *arbitrary*, it has an immediate meaning in terms of how energetic a CR should be to be flagged, with respect to the noise floor. We will detail in the following sections how our

approach provides a more consistent CRs flagging with respect to **calwf3**. Incidentally, this method of using the prime differences and flagging outliers is very similar to the one used by the `jump` step in the *JWST* pipeline (`https://github.com/spacetelescope/jwst/tree/master/jwst/jump`).

The CR detection step modifies the flow illustrated in Figure 1. A new loop is wrapped around the iterative loop illustrated in Section 4. Once the inner loop converges, a check for CRs is performed and if the CR loop itself halts, then the solution is found.

As before, we start from an initial guess of the flux using the measured counts, f^{old}. This is used for the first flagging of CR, simply by detecting outliers in the residuals with respect to this initial guess. The likelihood maximization algorithm then proceeds as before, but neglecting the CR affected intervals in the likelihood estimate. In particular, we drop the Poisson terms associated with flagged intervals. However, we keep the two Gaussian terms related to the probability of measuring a number of electrons (with readnoise), given the true number of electrons, i.e. the x_i values at the edges of the interval. The latter may contribute to the full likelihood unless an x_i is located between two flagged adjacent intervals, as it might be the case for two consecutive hits, or for CR hits that occur between m averaged frames in *JWST*-like detector mode of operations.

After finding the values of x_i that maximize the likelihood, limited to non CR hit intervals, we estimate f^{new} as in Figure 1 and we iterate until the condition $|f^{new} - f^{old}| < \epsilon$ is met. The f^{new} value is then used to check again which intervals may be affected by CRs. Changing the flux estimate changes both the variance, and thus the CR rejection threshold, as well as the reference value for calculating the residuals. If the CR affected intervals are estimated to be the same as in the previous iteration, we halt the algorithm, otherwise we optimize the joint probability again, this time using the newly flagged intervals. This more complete flow is illustrated in Figure 3.

6 Expression for the variance-covariance matrix

The expressions derived in this section are used in two distinct parts of our algorithm. First, the full variance-covariance matrix of the *true* flux estimates $f_i = \left\{\frac{\Delta x_{i,i+1}}{\Delta t_{i,i+1}}\right\}_{i=1,...N_{groups}-1}$ is used to properly compute the weighted average $\bar{f} = \langle f_i \rangle$. Second we use the standard deviation of the *observed* fluxes, σ_{ϕ_i}, to establish whether there are intervals for which the observed count rate estimate $\phi_i = \left\{\frac{\Delta y_{i,i+1}}{\Delta t_{i,i+1}}\right\}_{i=1,...N_{groups}-1}$ differ more than a set threshold from the average estimate $\bar{f}$. This is used for CRs identification.

We note that the index i above runs on the number of groups, rather than the number of averaged frames. The two quantities are identical for instrument like WFC3/IR, where no frames are averaged together to obtain the single group values. However the two differ, in general, in *JWST*-like detector operation. The average operation introduces new terms in the variance-covariance matrix. All the equations below remain valid for read sequences without averaged frames, and the limit of such case (e.g. for WFC3/IR) can be recovered by simply setting $m = 1$.

Data: $y_1 ... y_N$, measured counts; $t_1 ... t_N$, sampling times
Result: f, true flux
Initialization:
$x_i^{old} \leftarrow y_i$; $f^{old} = \langle \frac{x_{i+1}^{old} - x_i^{old}}{t_{i+1} - t_i} \rangle$;
$\mathcal{S}_{CR}^{old} = \{\}$; `/* Empty set of intervals containing CR hits */`
for $i = 1, N-1$ **do**
  $(f^{old}, t_{i+1} - t_i, \sigma_{ron}^2, \sigma_q^2) \rightarrow \sigma_i^2$; `/* Electrons variance in interval` i `*/`
  $e_{exp}^- = f^{old}(t_{i+1} - t_i)$, $e_{obs}^- = (y_{i+1} - y_i)\mathrm{Gain}$;
  if $\| \frac{e_{exp}^- - e_{obs}^-}{\sigma_i} \| \geq \delta$ **then** `/* If difference greater than threshold */`
    $\mathcal{S}_{CR}^{old} = \mathcal{S}_{CR}^{old} \cup \mathcal{I}_i$; `/* Mark interval` i `as hit by CR */`
  end
end
while $k < MaxCRiter$ **do**
  while $j < Maxiter$ **do**
    if $\mathcal{I}_i \notin \mathcal{S}_{CR}^{old}$ **then** `/* Use non-CR hit intervals */`
      $x^{new} = \arg\max_{\{x_i\}} P(\{y_i\}, \{x_i\} | f^{old})$;
      $f^{new} = \langle \frac{x_{i+1}^{new} - x_i^{new}}{t_{i+1} - t_i} \rangle$;
    end
    if $|f^{new} - f^{old}| < \epsilon$ **then**
      continue ; `/* Exit this while loop */`
    else
      $f^{old} = f^{new}$; $j = j + 1$;
    end
  end
  $\mathcal{S}_{CR}^{new} = \{\}$; `/* Flag CR hits using the new flux */`
  for $i = 1, N-1$ **do**
    $(f^{new}, t_{i+1} - t_i, \sigma_{ron}^2, \sigma_q^2) \rightarrow \sigma_i^2$
    $e_{exp}^- = f^{new}(t_{i+1} - t_i)$, $e_{obs}^- = (y_{i+1} - y_i)\mathrm{Gain}$;
    if $\| \frac{e_{exp}^- - e_{obs}^-}{\sigma_i} \| \geq \delta$ **then**
      $\mathcal{S}_{CR}^{new} = \mathcal{S}_{CR}^{new} \cup \mathcal{I}_i$
    end
  end
  if $\mathcal{S}_{CR}^{new} = \mathcal{S}_{CR}^{old}$ **then** `/* If the CR affected intervals are the same */`
    stop;
  else
    $\mathcal{S}_{CR}^{old} = \mathcal{S}_{CR}^{new}$; $k = k + 1$;
  end
end

Figure 3: Likelihood Maximization procedure, including CR hits flagging

6.1 Variance-covariance of the true flux estimates

We define $\mathbf{\Sigma}_f$, the variance-covariance matrix of $\{f_i\}_{i=1,...N_{groups}-1}$. We begin by introducing two variables, the group time for group i, t_i, and the accumulated charge up to group i, x_i. We also introduce the time for individual frames $t_{i,k}$ and the accumulated charge up to the (i,k) frame, $x_{i,k}$. The index i runs over the number of groups, N_g, while the index k runs over the number of frames averaged together within a group, m. The $t = 0$ instant corresponds to the start of the exposure. By definition:

$$t_i = \frac{1}{m}\sum_{k=1}^{m} t_{i,k} \tag{13}$$

$$x_i = \frac{1}{m}\sum_{k=1}^{m} x_{i,k} \tag{14}$$

I.e., the group time and the group charge are the means of the frames times and frame charges. From equation (14) it follows that:

$$\begin{aligned}
\sigma^2_{x_i} &= \frac{1}{m^2}\sum_{k=1}^{m}\sigma^2_{x_{i,k}} + \frac{2}{m^2}\sum_{k=1}^{m}\sum_{j=k+1}^{m}\mathrm{Cov}(x_{i,k}, x_{i,j}) \\
&= \frac{1}{m^2}\sum_{k=1}^{m} x_{i,k} + \frac{2}{m^2}\sum_{k=1}^{m}\sum_{j=k+1}^{m} x_{i,k} \\
&= \frac{1}{m}x_i + \frac{2}{m^2}\sum_{k=1}^{m}\sum_{j=k+1}^{m} x_{i,k}
\end{aligned} \tag{15}$$

This takes advantage of the fact that the $x_{i,k}$ are Poisson variables and that the charge accumulated for $t_2 > t_1$ "contains" the charge accumulated until t_1 as well as any random fluctuations associated to it, thus:

$$\begin{aligned}
\sigma^2_{x_{i,k}} &= x_{i,k} \\
\mathrm{Cov}(x_{i,k}, x_{i,j}) &= x_{i,k} \quad \text{for } i < j
\end{aligned}$$

If we have an estimate of the flux, $\hat{f}$, we can also set:

$$\begin{aligned}
x_{i,k} &\rightarrow \hat{f}t_{i,k} \\
x_i &\rightarrow \hat{f}t_i
\end{aligned}$$

And equation (15) can be rewritten as

$$\sigma^2_{x_i} = \frac{1}{m}\hat{f}t_i + \frac{2}{m^2}\hat{f}L_i \tag{16}$$

where we have introduced:

$$L_i = \sum_{1\le k<j\le m} t_{j,k} \tag{17}$$

and L stands for "lower triangle sum" since the sum on the right hand side of equation (17) can be seen as the sum over the lower triangle of a $m \times m$ matrix where the (j,k) element is equal to $t_{j,k}$.

Given the variance of x_i, we can compute the variance of the individual flux estimates, $f_i = \frac{\Delta x_{i,i+1}}{\Delta t_{i,i+1}} = \frac{x_{i+1}-x_i}{t_{i+1}-t_i}$. We can write:

$$\sigma^2_{f_i} = \frac{\sigma^2_{x_i} + \sigma^2_{x_{i+1}} - 2\,\mathrm{Cov}(x_i, x_{i+1})}{(\Delta t_{i,i+1})^2} \tag{18}$$

The covariance term can be rewritten as:

$$\begin{aligned} \mathrm{Cov}(x_i, x_{i+1}) &= \frac{1}{m^2}\,\mathrm{Cov}\left(\sum_{k=1}^{m} x_{i,k}, \sum_{j=1}^{m} x_{i+1,j}\right) \\ &= \frac{1}{m^2}\sum_{k=1}^{m}\sum_{j=1}^{m}\mathrm{Cov}(x_{i,k}, x_{i+1,j}) \end{aligned} \tag{19}$$

And using the fact that:

$$\mathrm{Cov}(x_{i,k}, x_{i+1,j}) = x_{i,k} \tag{20}$$

equation (19) becomes:

$$\begin{aligned} \mathrm{Cov}(x_i, x_{i+1}) &= \frac{1}{m^2}\sum_{k=1}^{m}\sum_{j=1}^{m} x_{i,k} \\ &= x_i \end{aligned} \tag{21}$$

Putting everything together, we have that

$$\begin{aligned} \sigma^2_{f_i} &= \frac{\sigma^2_{x_i} + \sigma^2_{x_{i+1}} - 2x_i}{(\Delta t_{i,i+1})^2} \\ &= \frac{\hat{f}}{(\Delta t_{i,i+1})^2}\left[\frac{t_i + t_{i+1}}{m} + \frac{2(L_i + L_{i+1})}{m^2} - 2t_i\right] \end{aligned} \tag{22}$$

The missing pieces to complete the expression of $\mathbf{\Sigma}_f$ are the off-diagonal terms $\mathrm{Cov}(f_i, f_j)$, $i \neq j$. They can be expressed as:

$$\begin{aligned} \mathrm{Cov}(f_i, f_j) &= \frac{\mathrm{Cov}(x_{i+1} - x_i, x_{j+1} - x_j)}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \\ &= \frac{\mathrm{Cov}(x_{i+1}, x_{j+1}) - \mathrm{Cov}(x_i, x_{j+1}) - \mathrm{Cov}(x_{i+1}, x_j) + \mathrm{Cov}(x_i, x_j)}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \end{aligned} \tag{23}$$

Using the fact that:

$$\mathrm{Cov}(x_i, x_j) = x_i \quad \text{if } i < j \tag{24}$$

which can be derived in the same way equation (21) was derived, and assuming that $i < j$, equation (23) becomes:

$$\begin{aligned}
\mathrm{Cov}(f_i, f_j) &= \frac{x_{i+1} - x_i - \mathrm{Cov}(x_{i+1}, x_j) + x_i}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \\
&= \frac{1}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \times \begin{cases} x_{i+1} - \sigma^2_{x_{i+1}}, & \text{if } j = i+1. \\ 0, & \text{otherwise.} \end{cases} \\
&= \frac{\hat{f}}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \times \begin{cases} t_{i+1}(1 - \frac{1}{m}) - \frac{2}{m^2}L_{i+1}, & \text{if } j = i+1. \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{25}$$

Therefore only in the case of consecutive intervals, $j = i + 1$, the flux estimates f_i have non-zero covariance. This is equivalent to saying that the variance-covariance matrix, $\mathbf{\Sigma}_f$ has only the major and first minor diagonals with non zero terms, i.e:

$$\mathbf{\Sigma}_f \rightarrow \frac{\hat{f}}{\Delta t_{i,i+1}\Delta t_{j,j+1}} \times \begin{cases} \frac{t_i + t_{i+1}}{m} + \frac{2(L_i + L_{i+1})}{m^2} - 2t_i, & \text{if } i = j. \\ t_{i+1}(1 - \frac{1}{m}) - \frac{2L_{i+1}}{m^2}, & \text{if } |i - j| = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

6.2 Standard deviation of the observed flux rate

In order to estimate whether a CR might have affected a ramp in the time between the group averaged times, t_i and t_{i+1}, we compare the difference between the observed rate, $\phi_i = \frac{\Delta y_{i,i+1}}{\Delta t_{i,i+1}}$ and the current estimate of the true flux, $\hat{f}$ at each iteration. If the absolute difference is greater than a fixed threshold times σ_{ϕ_i}, the standard deviation for ϕ_i, the interval is discarded from the next iteration. The σ_{ϕ_i}'s contain contributions from the photon noise (from the x_i), from the readnoise, and from the bit-shift and quantization noise. These three separate contributions are summed in quadrature:

$$\sigma_{\phi_i} = \sqrt{\sigma^2_{phot,\phi_i} + \sigma^2_{ron,\phi_i} + \sigma^2_{bq,\phi_i}} \tag{27}$$

The first term can be immediately derived from the $i - th$ diagonal term of equation 26. The second term is given by:

$$\sigma^2_{ron,\phi_i} = \frac{2}{m}\sigma^2_{ron} \tag{28}$$

Where the factor of 2 comes from the fact that we are taking the difference of two groups, and the factor $\frac{1}{m}$ comes from averaging multiple frames together.

The third term is the variance of a uniform distribution:

$$\sigma^2_{bq,\phi_i} = \frac{2}{12}(gm)^2 \tag{29}$$

Where the 2 factor again indicates that there are two separate quantization and bit shift operations. The bit-shift operation effectively rounds numbers by losing information on the

remainder of the integer division by m, thus digital numbers differing by less than m are not distinguishable. In the case of $m = 1$ this term is still valid, but its interpretation is simply that of a rounding error when converting an analog signal to a digital number. The multiplication by the gain, g correctly ensures that this term is in units of electrons, consistently with the other two variance terms.

7 Algorithm implementation details

In this section we highlight some more detailed specifics of the actual algorithms. The corresponding python code is available at `https://github.com/spacetelescope/Up_the_ramp`. In particular the fitting algorithm is contained within the `ramp_utils/fitter.py` module. Other code in the github repository includes the `ramp_utils/ramp.py` module, used for creating simulated ramps, and a suite of python notebooks that can be used to run simulations.

7.1 Initial guess

The values of the true number of photo-generated electrons (i.e. before the electronics introduce any readnoise), x^{old} are initially set to the measured counts values multiplied by the gain. An adjustment is performed to ensure that the numbers in the initial guess are never decreasing, i.e. $x_i^{old} \leq x_{i+1}^{old}$. This is achieved by simply scanning the x_i^{old}'s in ascending order and if $x_{i+1}^{old} < x_i^{old}$ we set $x_{i+i}^{old} = x_i^{old}$.

Note that the measured counts may be decreasing, due to the specifics of the readnoise realizations; for example imagine a true flux of zero and two consecutive frames with readnoise realizations $r_{i+1} = -r_i$, one would have a negative measured countrate (in electrons per second) of $-2r_i/(t_{i+1} - t_i)$. The true number of photo-generated electrons in that interval is however equal to 0. We remind the reader that we are using the auxiliary x variables to estimate the true flux, f, by exploiting the fact that $P(\Delta x_{i,i+1}|f\Delta t_{i,i+1}) \sim Poi(f\Delta t_{i,i+1})$. This Poisson distribution is only valid for $\Delta x_{i,i+1} \geq 0$.

We will return on the question of negative fluxes in Section 10.3, but let us digress briefly here. Using the $-2r_i/(t_{i+1} - t_i)$ value as an estimate of the true flux leads to the *absurd* result of a negative flux. The problem is that there are two random processes at play, the Poisson generation of photo-electrons and the Gaussian readnoise. Using $-2r_i/(t_{i+1} - t_i)$ equates assuming that both processes are Gaussian. In that case using the measured value as an approximation of the true value would be the correct thing to do. Properly combining the Poisson probability for the generation of photo-electrons and the Gaussian readnoise does not lead to negative estimated fluxes, as we show in section 10.3.

7.2 Electrons vs. counts: the gain

We reiterate that the actual output of an IR detector is a sequence of digitized numbers, i.e. counts. We transform back those integer counts into electrons by dividing the counts by the gain and then we use the `floor` operation to turn the electron numbers into integers. Using electrons rather than counts ensures that we can properly treat the Δx quantities as

Poisson. While incident photon and photo-generated electrons are distributed as Poisson variables, counts are not because if $X \sim Poi(\lambda)$ and $Y = aX$ then $Y \nsim Poi(a\lambda)$.

7.3 Flux average

At initialization we set the value of the flux to the median of the individual flux estimates, $\bar{f} = \langle f_i \rangle = \langle \frac{\Delta x_{i,i+1}}{\Delta t_{i,i+1}} \rangle$. After experimenting we have found that using the median rather than the mean helps with convergence, in particular when few groups are available and CR hits are present. Since an estimate of the flux is needed to compute the standard deviation of the electron rate that is in turn used to flag intervals where CRs hits may happen, the fact that the median is more outlier-resistant than the mean allows and earlier flagging of CR hits.

In the subsequent iterations the average electron rate is estimated by taking into account $\mathbf{\Sigma}_f$, the full variance-covariance matrix of the $f_i = \left\{ \frac{\Delta x_{i,i+1}}{\Delta t_{i,i+1}} \right\}_{i=1,\ldots N_{groups}-1}$ which was calculated in section 6.1; for ease of reading we drop the f subscript in the following. We first restrict $\mathbf{\Sigma}$ to only the intervals that are not affected by CRs, i.e. $\mathbf{\Sigma} \rightarrow \mathbf{\Sigma}_{\overline{CR}}$, where $\overline{CR}$ indicates absence of CR-affected intervals. We then use the expression for the weighted mean of correlated variables to obtain the current estimate of the flux:

$$\bar{f} = \sigma_f^2 (\mathbf{J}^T \mathbf{W} \mathbf{F}) \tag{30}$$

where:

$$\begin{aligned} \mathbf{W} &= \mathbf{\Sigma}_{\overline{CR}}{}^{-1} \\ \sigma_f^2 &= (\mathbf{J}^T \mathbf{W} \mathbf{J})^{-1} \\ \mathbf{F} &= [f_1, \ldots, f_{N_{groups,\overline{CR}}-1}]^T \\ \mathbf{J} &= [1, \ldots, 1]^T (\text{of length } N_{groups,\overline{CR}} - 1) \end{aligned}$$

$N_{groups,\overline{CR}}$ indicates the number of groups that are not affected by CR hits, and is also the rank of the $\mathbf{\Sigma}_{\overline{CR}}$ matrix.

This estimate of the flux is used to set the new value of the flux, $f_{new} \rightarrow \bar{f}$. The latter is compared to the flux estimate from the previous iteration, f_{old} and in case convergence is not reached, f_{new} is used to compute the likelihood in the next iteration.

7.4 Likelihood Minimization

At each iteration, equation (9), or its generalization to the case of averaged frames equation (12), are maximized with respect to $\vec{x}$, using the value of the estimated flux at the beginning of that iteration, f^{old}. In practice we minimize the logarithm of the same expression.

We adopt the Nelder-Mead algorithm, in its implementation within the `scipy.optimize` package. The Nelder-Mead method is a direct search method, particularly suitable to highly dimensional problems, and specially problems for which the derivatives of the function to minimize cannot be computed (Nelder & Mead, 1965).

If the minimization step fails to converge, we give it slightly different initial conditions and attempt again. If we fail a number of times greater than *conv_attempts*, we then set

$x^{new} \rightarrow x^{old} + \varepsilon$, where the ε are drawn from a normal distribution with variance given by the readnoise variance, i.e $\varepsilon \sim \mathcal{N}(0, \sigma^2_{ron})$. These x^{new} are then forced to be non-decreasing by scanning their values and if $x^{new}_{i+1} < x^{new}_i$ we set $x^{new}_{i+1} = x^{new}_i$. *conv_attempts* is set by default to 5 but can be changed by the user.

7.5 Thresholds and stoppage

Two separate criteria need to be satisfied for our fitting procedure to stop: the CR hits must be identified and the flux estimation must converge.

7.5.1 Convergence of the cosmic rays finding algorithm

Our first criterion for convergence is that there must be no change in the set of intervals flagged as hit by CRs. After each fitting iteration is complete and a new value of the flux is estimated, we use that value to compute the expected standard deviation of the observed counts in each sampling interval, as described in Section 6.2. We then compare the expected versus observed photo-electrons and check whetehr the difference is within δ times the standard deviation, see Section 5.2. This allows us to flag a set $\mathcal{S}^{new}_{CR} = \mathcal{I}_{h1}, ..., \mathcal{I}_{hN}$ where the hN intervals are flagged as hit by CRs. If $\mathcal{S}^{new}_{CR} = \mathcal{S}^{old}_{CR}$ the CR flagging algorithm is considered converged.

We introduce *maxCRiter*, the maximum number of iterations for accepting a change in the set of CR hit intervals identification. *maxCRiter* is set by default to 10. In our experiments the CR flagging converges rapidly and we never found it necessary to interrupt the fitting process using such parameter.

7.5.2 Convergence of the flux estimation algorithm

The flux estimation iterative algorithm stops when either one of these conditions is met:

- $|f^{new} - f^{old}| < \epsilon$
- iterations $>$ *maxiter*

The convergence threshold, ϵ, is tunable and it is set by default to $\frac{g}{t_N - t_1}$, i.e. the gain divided by the time between the first and last group (this can in principle slightly differ from the exposure time if multiple frames are averaged together). This threshold is the minimum flux difference that can be appreciated given the exposure time. Two flux values that differ by less than ϵ cannot produce a measurable count rate difference, on average. If the algorithm fails to find a stable solution before *maxiter* iterations, it exits with an error. *maxiter* is set by default to 50 but can be changed by the user.

7.6 Goodness of fit

We implement a goodness of fit test to assess the quality of our fitted values. In the following, we indicate the values after convergence with a $\tilde{.}$ symbol. We first use equation (12) to

compute a reference probability value:

$$P_{ref} = \prod_{i=1}^{N} \mathcal{N}(y_i - \tilde{z}_i; \frac{\sigma_{ron}^2}{m}) \times \prod_{i=1}^{m\times N-1} Poi(\Delta\tilde{x}_{i,i+1}; \tilde{f}\Delta t_{i,i+1}) \tag{31}$$

Thus P_{ref} represents the probability of observing the measured y_i and having true values of the accumulated photo-electrons $\tilde{x}_i$, if the true flux is equal to the fitted flux, $\tilde{f}$. We use $\tilde{f}$ to produce a set $\mathcal{T}$ of simulated photo-electrons for each of the read intervals, $\Delta x'_{i,i+1}$, by drawing random numbers from Poisson distributions with $\lambda = \tilde{f}\Delta t_{i,i+1}$ (the $'$ symbol marks simulated values). We further generate a set of $\mathcal{T}$ realizations of the readnoise for each of the reads, x'_i thus obtaining $y'_i = x'_i + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \frac{\sigma_{ron}^2}{m})$. We use again equation (12) to compute probabilities:

$$P'_j = \prod_{i=1}^{N} \mathcal{N}(y'_i - z'_i; \frac{\sigma_{ron}^2}{m}) \times \prod_{i=1}^{m\times N-1} Poi(\Delta x'_{i,i+1}; \tilde{f}\Delta t_{i,i+1}) \tag{32}$$

The $\left\{P'_j\right\}_{j=1,..,\mathcal{T}}$ represent the probabilities of observing values y'_i and generating x'_i given the true flux, $\tilde{f}$ and the readnoise variance, and are thus representative of the type of probability values we should be expecting for P_{ref}.

Similarly to the definition of the p-value for a traditional χ^2 test, we define our own p-value as:

$$p = \frac{\sum_{j=1}^{\mathcal{T}} H(P_{ref} - P'_j)}{\mathcal{T}} \tag{33}$$

where $H(x)$ is the Heaviside function:

$$H(x) = \begin{cases} 1, & \text{if} \quad x > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

By definition $0 < p < 1$. We are therefore counting the fraction of times that the best-fit solution gives a higher probability with respect to random draws that come from and underlying distribution based on the best-fit result. The p-value tells us how probable our actual solution for the $\tilde{x}_i$ given $\tilde{f}$ is, relative to random realizations of both the Poisson process for the photons given the same true flux and the Gaussian process for the readnoise, given the same readnoise variance. It is obviously possible that by chance individual values of the p-value for single ramps fits may be very low. This is e.g. the case when two consecutive reads have first very high and then very low realizations of the readnoise. The p-value thus defined is mostly useful to verify ensemble properties, using multiple ramps, for example corresponding to multiple pixels on an image. In such cases it is possible to flag unexpected behaviors globally and not individually.

For example, we make use of the p-value in section 10.5, where we test what happens if the underlying incident flux is not constant with time. We show that although in such situation the average flux can be well recovered, in truth the flux variations within the ramp make the solution not a good fit overall. This manifests itself as very low p-values over an ensemble of ramps.

8 Simulating ramps

Our code has been validated by extensive testing using simulated ramps with known input fluxes, known CRs events, etc. The code for simulating the ramp measurements can be found under the main repository `https://github.com/spacetelescope/Up_the_ramp`, in the `ramp_utils/ramp.py` module which provides two classes: `RampTimeSeq` and `RampMeasurement`.

8.1 The `RampTimeSeq` class

The `RampTimeSeq` class is used to generate the frames timing sequences. It has default hardcoded sequences for the WFC3/IR instrument, but can be used to generate `GENERIC` timing sequences with arbitrary frame times, number of groups, number of averaged frames, number of skipped frames. This class also computes the sum L_i of equation (17) which only depends on the frame times. Section A.1 in the Appendix shows some example use cases of this class.

8.2 The `RampMeasurement` class

The `RampMeasurement` is the workhorse class for simulating ramps. It takes an instance of the `RampTimeSeq` class as input as well as a value of the flux, to generate an observed ramp in units of counts. Optional parameters are the gain, the standard deviation of the readout noise, the KTC noise, the bias level, and the full well capacity. The latter is in fact not utilized as we have not implemented a prescription for dealing with saturated ramps yet. The optional parameters are required for `GENERIC` timing sequences, but are instead filled up with default values for `HST/WFC3/IR` timing sequences. A further optional parameter is a set of CR events. These are specified as a dictionary of hit times and hit deposited counts. The `RampMeasurement` class has a method `add_background` that can be used to add a variable background a posteriori, i.e. on top of the "source" counts. Finally, also this class has a convenience plotting method to display its results.

Section A.2 in the Appendix shows examples of using the `RampMeasurement` class to generate a ramp and, if required, add CR hits. Section A.3 in the Appendix shows how to use the `RampMeasurement.add_background` method. The latter is heavily used in Section 10.5 to test the behavior of our method and of **calwf3** when the assumption of constant flux fails due to the presence of a time-varying background.

9 Fitting ramps

This section illustrates the general functionalities of our fitting routine. We start from a simulated ramp with:

- Frame time of 10 seconds
- 2 averaged, 4 skipped frames per group, 8 groups
- input flux of 1 electrons per second

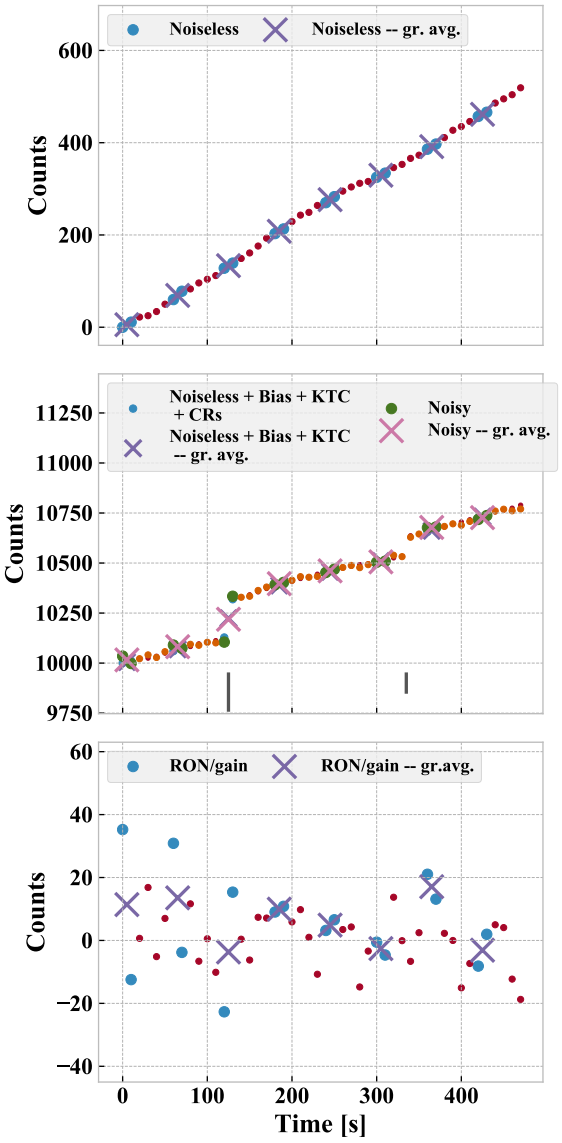

Figure 4: Simulated ramp used for illustration of the fitting method. In all the panels large dots correspond to averaged frames, small dots to skipped frames and crosses to the group averages. **Top:** "noiseless counts", i.e. photo-generated electrons divided by the gain. These values contain the random Poisson fluctuations in the photon numbers, but no measurement errors (hence noiseless). **Center:** noisy counts. The readnoise and CR deposited energies are added to the noiseless counts. On-chip average is performed. A single value of the bias and of the initial random KTC noise is added to the ramp. These two noise sources matter only as offsets and do not affect the recovery of the true flux. The vertical gray lines indicate the CR hits times and their heights correspond to the deposited counts. It is worth noting that by design these 2 CR hits happen between two of the averaged frames (first hit) and between two of the skipped frames (second hit). The consequences are different. In the first case 2 intervals are adversely affected, in the second case just one. **Bottom:** actual realizations of the readnoise for the individual frames. The averaged read noise per group is also shown, showcasing the reduction is variance of the effective readnoise by a factor of $1/m$ (in this case $m = 2$)

- gain of 1 electron/ADU and readout noise $\sigma_{ron} = 15$ electrons
- 2 CRs hits, one between two of the averaged frames, one between skipped frames

This ramp is shown in Figure 4.

9.1 The `IterativeFitter` class

The code that implements the iterative maximum likelihood approach detailed in all the preceding sections is available at at `https://github.com/spacetelescope/Up_the_ramp`, under the `ramp_utils/fitter.py` module. This module defines the `IterativeFitter` class. This class is initialized by passing it a `RampMeasurement` object. The `perform_fit` method is the workhorse method that implements the core algorithms for the fit, calls the like-

lihood computation and minimization, advances the iterations, updates the estimates of the variance-covariance matrix, identifies CRs and checks for convergence. Convenience methods are available to plot the fit results (`test_plot`) and check the goodness of the fit (`goodness_of_fit`).

The `perform_fit` method does not return the fitted flux value, which however is stored in the `mean_electron_rate` attribute. The variance of the estimated flux is stored in the `sigmasq_h` attribute. The returned values are (`error, counter, good_intervals, crloops_counter`) which in order represent: an error code that can be used as diagnostic in case of non-convergence, the number of iterations before convergence, the set of intervals considered as hit/not-hit by a CR, and the number of iterations used to identify CRs.

The example in Listing 1 illustrates how to use the `IterativeFitter` class and Figure 5 shows the results when the code is applied to the ramp of Figure 4.

```
import numpy as np
from ramp_utils.ramp import RampTimeSeq,RampMeasurement
from ramp_utils.fitter import IterativeFitter

# The ramp time sequence, in this case a generic ramp
dt, nf, ns, ng = 10, 2, 4, 8
myramp = RampTimeSeq('GENERIC', ng, nframes=nf, nskips=ns, read_times=dt*
    np.arange(ng*(nf+ns)))

# Input values
myflux = 1
CRdict = {'times':[125.,335.],'counts':[180,90]}

# The measurement object
# (shown in Figure 4)
mymeas = RampMeasurement(myramp,myflux,gain=1.,RON=15.,KTC=0,bias=10000,
    full_well=100000,CRdict=CRdict)

# Initialize the fitter
myfitter = IterativeFitter(mymeas,fitpars={'one_iteration_method':'Nelder-
    Mead'})

# Perforn a fit (use a CR rejection threshold of 4)
err, count, gi, crlcount = myfitter.perform_fit(CRthr=4)

#Plot the fit results shown in Figure 5
myfitter.test_plot()

#Print some diagnostics
print('fitted flux (e/s)', myfitter.mean_electron_rate) # Fit output
print(gi)   # Intervals flagged as containing CRs
print('Error message',err)  # Error message used to debug fit failures
print('Iterations',count)   # Number of iterations for success
print('Iterations for CR rej',crlcount) # Number of CR rejection
    iterations

#Print the goodness of fit measure
myfitter.goodness_of_fit(mode='full-likelihood')
print('Statistic:',myfitter.gof_stat)
```

```
print('p-value:',myfitter.gof_pval)
```

Listing 1: An example using the `IterativeFitter` class

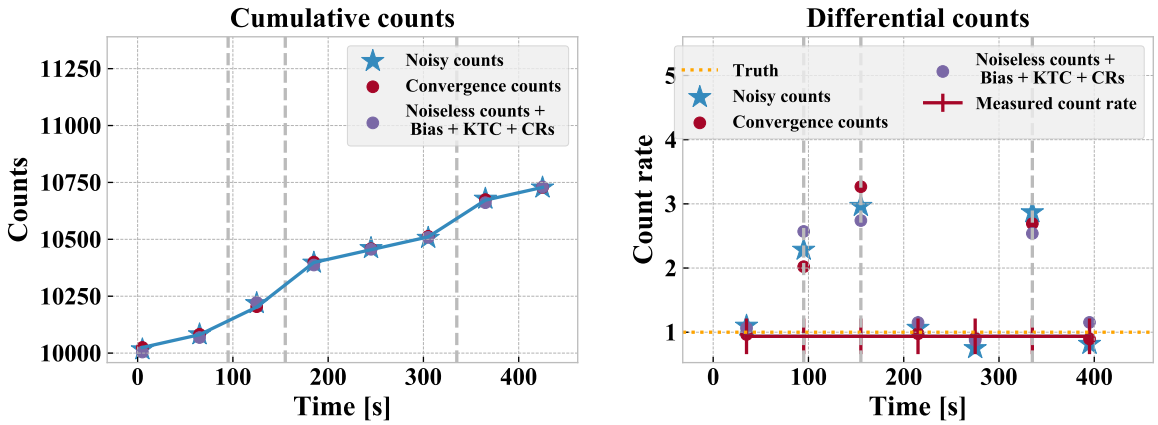

Figure 5: Results of the fitting method when run on the ramp of Figure 4. CRs are correctly identified and flagged, and are marked as vertical dashed lines. The input flux is well recovered, as shown by the red line in the right panel. Noisy counts are the measurements for each group, y_i, while convergence counts are the z_i values, i.e. the group-average of the estimated photoelectrons per frame, $x_{i,j}$, divided by the gain. The "noiseless" counts in purple circles are the true values of the $x_{i,j}$, i.e the photoelectrons withouth readnoise, adjusted by the constant bias and KTC noise, as well as by the CR deposited energy. In a heuristic sense, the red circles, convergence counts, should converge to the purple circles. On the left panel, the blue line is the fit result, obtained by using the best fit estimated rate and adding the estimated deposited energy by each CR. The latter are in turn estimated by using the z_i estimates before and after the identified hits.

10 Comparison with calwf3

We simulated WFC3/IR ramps for a range of read sequence types and the input fluxes. The input fluxes are chosen to give us SNR covering a variety of regimes. For each combination of inputs we generate 10,000 ramps.

In order to compare our results to the performance of the **calwf3** pipeline, we format the measured counts from the simulated `RampMeasurement` objects into `.fits` files that can be handled by **calwf3**. The `RampMeasurement` object provides measurements in digital units, or counts. This corresponds to the units in the *raw* files that are the inputs of the **calwf3** processing. However our simulations do not include several non-ideal effects that a typical-usage **calwf3** run must account for, including detector non-linearity, dark current, flat fielding, bias subtraction. For this reason we set most of the **calwf3** switches to `OMIT` which effectively skips these steps. The step of the pipeline we are interested in testing is in fact `CRCORR` which is set to `PERFORM`.

10.1 Behavior in different signal-to-noise ratio regimes

The first test we conducted consisted in using a fixed read pattern and changing the input flux to span from very low, readnoise dominated SNR regime, to very high, Poisson dominated SNR regime. We used the STEP200 WFC3/IR read sequence, with 13 samples up the ramp, for an exposure time of 999.231 s (see the WFC3 Instrument Handbook, Dressel, 2019). Note that the Instrument Handbook use a 0-based counting scheme for the samples up the ramp, thus NSAMPLES = 12 corresponds to 13 samples. This is due to the fact that in fitting the cumulative counts (as **calwf3** does), the 0-th samples can be subtracted from all subsequent ones without loss of information on the slope, the 0-th sample being in fact the intercept of the linear run of counts with time. Since we use differential counts, the 0-th sample is in fact useful in establishing the rate in the first interval (between the 0-th and 1-st frames).

We randomly add CR hits to the ramps by drawing hits from a Poisson distribution with a CR hit rate per pixel of $5e^{-4}s^{-1}$, i.e. one hit every 2000 seconds on average. With typical durations of the ramps of $\sim$ 1000 seconds, this lets us have about one half of the ramps being hit by a CR. Since the probability of a CR event is Poissonian, we also have a tail of ramps experiencing multiple hits. We note that this rate is higher than the true rate of CR that the WFC3/IR detector is typically exposed to which is of the order of $3e^{-5}s^{-1}$, or 15 times smaller than what we used in our simulations (see Barker et al., 2009).

Table 1 shows the range of inputs we used and the corresponding fit results for both our method and **calwf3**. From the input fluxes and the exposure time of 999.231 s, it is possible to see that the total number of photo-electrons per ramp goes from ~ 9 to $\sim 62,500$. In the former case the noise is strongly dominated by the Gaussian readnoise, while in the latter the Poissonian photon noise completely dominates. The table reports the actual number of simulated CRs, as well as the number of detected ones (Det), the number of false positives (FP), and for ease of reading we explicitly report the missed ones or false negatives (FN), that in fact are equal to total minus detected hits. The output mean flux and standard deviation over the sample of 10,000 simulations are also shown in 3 variants, reported in 3 separate column blocks in Table 1. In the first case all ramps are counted. In the second, we only display results for ramps that are not affected by CRs. In the third we only considered CR-affected ramps.

The most striking difference between the two approaches is that the `IterativeFitter` class does a much better job at detecting CRs. The number of false negatives for **calwf3** is considerably higher. Our method tends however to flag a larger number of spurious CR hits. We will return on this subject in Section 10.4. Given that **calwf3** misses a large number of CRs, the statistics are strongly affected. Missed CRs manifest themselves as "extra flux" resulting in a long tail at high values of the fitted flux. This tail, in turn, biases the mean estimate of the flux towards high fluxes and increases the estimates of the standard deviation. This can be seen both in the "All ramps" and the "YES CR ramps" columns of Table 1.

If we limit ourselves to the "No CR ramps" column, we can see that **calwf3** and our approach give very similar results, both in the mean and the standard deviation of the output. This is true at least for fluxes greater than 0.1 (corresponding to SNR$\sim$8.5 in this case). At lower fluxes **calwf3** mean output flux is more similar to the input value than for our fitter, and we dedicate Section 10.3 to explaining why this is an expected result of using the full Poisson-times-Gaussian likelihood versus the *de facto* pure-Gaussian approximation

Inputs		Fitter	CR outputs			All ramps		No CR ramps		Yes CR ramps	
Flux	CR#		Det	FP	FN	μ	σ	μ	σ	μ	σ
0.009	4874	This work	4680	297	194	0.030	0.033	0.017	0.015	0.051	0.042
		calwf3	2874	4	2000	0.089	0.282	0.009	0.013	0.216	0.422
0.025	4959	This work	4739	106	220	0.039	0.034	0.026	0.015	0.059	0.044
		calwf3	2852	5	2107	0.110	0.256	0.025	0.014	0.239	0.372
0.1	5060	This work	4877	41	183	0.094	0.035	0.089	0.022	0.102	0.046
		calwf3	3001	6	2059	0.185	0.302	0.100	0.017	0.316	0.448
2.5	4871	This work	4610	42	261	2.498	0.066	2.496	0.058	2.501	0.078
		calwf3	2805	4	2066	2.587	0.281	2.500	0.053	2.722	0.409
25.0	4999	This work	4361	30	638	25.012	0.189	24.998	0.159	25.034	0.225
		calwf3	2619	5	2380	25.123	0.397	24.999	0.160	25.318	0.551
62.5	4929	This work	3972	25	957	62.531	0.291	62.506	0.254	62.570	0.338
		calwf3	2227	2	2702	62.676	0.498	62.507	0.254	62.942	0.647

Table 1: Input and output values of our comparison tests between **calwf3** and the `IterativeFitter` approach of this paper. In this case we explored the pipelines behaviors for different SNR regimes. In all the simulations reported here we adopt a STEP200, NSAMP=12 WFC3/IR read sequence, resulting in a total exposure time of 999.231s. The input flux and output μ and σ are all in electrons per second. CR# indicates the number of CR hits that are added to the ramps. Under CR outputs we list the number of detected CRs (Det), the number of false positives (FP) and the number of missed, or false negative CR hits (FN). All / No CR / Yes CR ramps indicate results for all the 10,000 simulated ramps, only those not affected by CR hits and only those affected by CR hits, respectively.

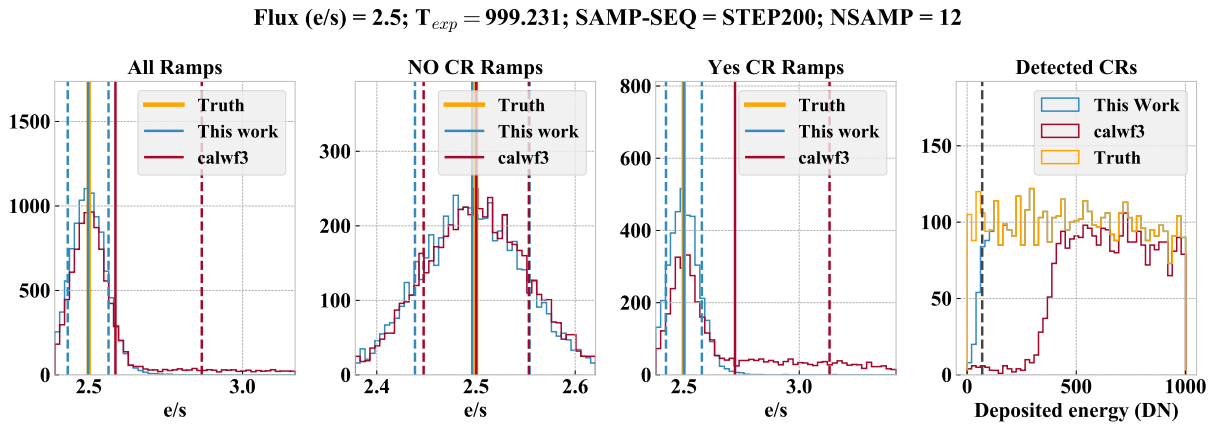

Figure 6: Results for a SPARS200, NSAMP=12 sequence, with input flux of 2.5 electrons per second, resulting in a SNR of ~ 50. Results are shown for all ramps, as well as for ramps with and without CRs separately. The histogram of all results over 10,000 runs, as well as the means (solid lines) and the 1σ limits (dashed lines) are shown for both our method and **calwf3**. The right panel shows the distribution of input CR deposited counts and the recovered distribution from **calwf3** and our `Iterative Fitter` class. The vertical dashed line in the right panel shows the threshold up to which we predict to be able to recover CR hits, in perfect agreement with the observed output.

of **calwf3**, which uses a least-squares estimator to retrieve the true flux.

Figure 6 shows the output of a test, using a SPARS200, NSAMP=12 sequence, with input flux of 2.5 electrons per second, resulting in a SNR of ~ 50. We show the results for all the tests, and then separately for ramps with and without CR hits. We also show the histogram of input CR deposited energies and the fraction of those recovered. The CR hits histogram also shows a dashed vertical line of our expectation for the threshold of detectable CR hits, given the input flux, the sample sequence and the detector characteristics, according to the equations of Section 6.2. We show that our ability to find CRs is predictable and tunable, i.e. our actual detection limit falls where expected in terms of deposited CR energy.

10.2 Behavior at fixed signal-to-noise but changing read sequence

In this Section we report on a different type of test. While in Section 10.1 we kept the read sequence fixed and changed the input flux, here we keep the input flux and the total exposure time constant, effectively keeping the number of incident photons constant. We however changed the underlying read sequence. The WFC3/IR predefined sequences (see the WFC3 Instrument Handbook, Dressel, 2019), have some amount of redundancy in terms of total exposure time. In particular it is possible to use the SPARS50/SPARS100/SPARS200 sequences to achieve a total exposure time of 602 seconds. What changes is the number of samples, or intervals used. By changing only the number of reads we can both test the impact of changing the effective readnoise as well as the effects on CR detection ability.

To assess the effects of changing the readnoise we conduct experiments at very low incident flux: 0.04 electrons per second. This corresponds to about 25 total electrons and

Inputs			Fitter	CR outputs			All ramps		No CR ramps		Yes CR ramps	
Flux	Sequence	CR#		Det	FP	FN	μ	σ	μ	σ	μ	σ
0.04	SPARS50 NSAMP13	3059	This work	2934	6	125	0.079	0.044	0.075	0.039	0.092	0.055
			calwf3	1924	0	1135	0.097	0.226	0.040	0.024	0.256	0.398
	SPARS100 NSAMP7	3062	This work	2929	31	133	0.074	0.080	0.072	0.043	0.080	0.138
			calwf3	1852	3	1210	0.118	0.290	0.040	0.029	0.338	0.499
	SPARS200 NSAMP4	2974	This work	2751	3230	223	0.320	0.954	0.056	0.075	1.154	1.685
			calwf3	1590	119	1384	0.223	0.879	0.040	0.034	0.746	1.617
40.00	SPARS50 NSAMP13	3007	This work	2740	12	267	40.003	0.266	39.999	0.260	40.017	0.284
			calwf3	1719	0	1288	40.093	0.410	39.999	0.261	40.364	0.597
	SPARS100 NSAMP7	2976	This work	2602	14	374	40.009	0.291	40.001	0.258	40.034	0.367
			calwf3	1644	1	1332	40.116	0.465	40.001	0.259	40.447	0.704
	SPARS200 NSAMP4	2968	This work	2029	1181	939	40.469	2.162	40.001	0.261	41.815	3.935
			calwf3	1367	80	1601	40.220	0.882	40.002	0.260	40.851	1.511

Table 2: Effects on the fit output of changing the read sequence at fixed incident flux and exposure time. Two cases are shown, for two regimes of SNR.

an SNR between 2.5 and 4 depending on the read sequence, whereas the pure photon SNR would be 5. On the other hand, to verify the impact of CRs we use a much higher flux of 40 electrons per second, corresponding to an SNR of about 155. Results for both cases are shown in Table 2.

For the low incident flux we can focus on the "NO CR ramps" columns of the table. As seen in Section 10.1, **calwf3** does a better job at recovering the mean flux, although the reader should refer to Section 10.3 for the details. For both pipelines we observe the expected increase in variance as the number of reads decreases, an effect of the increasing effective readnoise. We also see that our approach improves in the mean as the number of reads decreases. We interpret this apparently counterintuitive result in Section 10.3 as well.

At high SNR it is instructive to focus on the "YES CR ramps" columns of Table 2, where the effects of CRs are more explicit. We see again that our approach does a much better job at detecting CRs while allowing for a slightly larger number of false positives. Overall we see a trend in a larger number of both false positives and false negatives as the number of reads decreases for both pipelines. Our pipeline seems more sensitive to working in a low number of samples regime than **calfw3** in that the worsening of its behavior is faster. Nevertheless the number of detections is always higher for our pipeline with respect to **calwf3**, although the number of false positives explodes. This is due to the fact that the mean flux we estimate from a weighted average of the individual interval's rates at each iteration (see equation 30) is strongly affected by outliers. A mitigation strategy could be to jointly compute a more outlier-resistant average estimate, such as the median or a σ-clipped average, and adopt the latter for CR hit identification, while the former is the most unbiased estimate once the CR hits have been identified.

Another global trend we observe in the high SNR case is the increase of the fit results standard deviation as the number of reads decreases. This is explained as a larger fractional decrease in effective exposure time as larger chunk of the nominal exposure time is lost to CRs when the affected intervals are longer.

10.3 Results for $f \to 0$

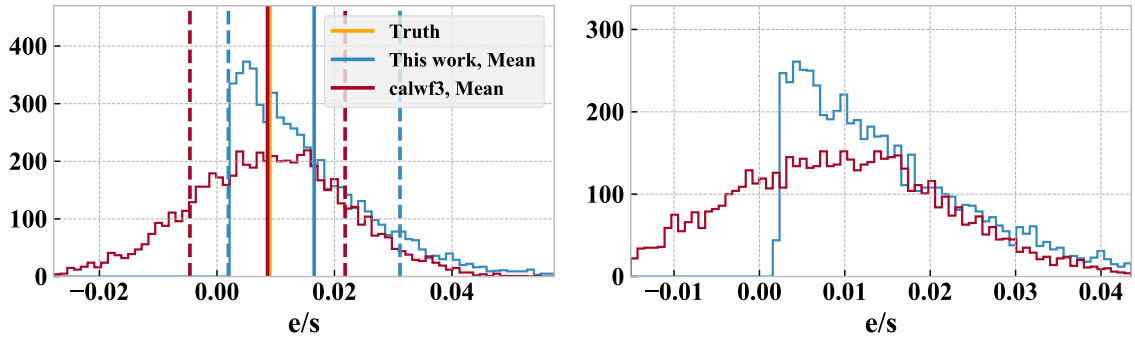

Figure 7: Results for a SPARS200, NSAMP=12 sequence, with input flux of 0.009 electrons per second, resulting in a SNR of $\sim$ 1.5. The figure shows the output distribution for the "NO CR ramps", i.e. ramps that are not affected by CRs, to better highlight the intrinsic effects of using a Poisson-Gaussian likelihood rather than a pure-Gaussian one, without the extra variance and structure added by the CR hits and their false positive and false negative detections. The right panel is a zoomed-in version of the left one that allows to better appreciate the tail at negative values for the **calwf3** fits, and the longer tail at positive values for our method.

Sections 10.1 and 10.2 both show how in the low SNR regime the **calwf3** recovers the mean input flux well, while our pipeline does not. Although this is a desirable outcome of **calwf3**, it comes at a cost. Figure 7 highlights such tradeoff. The figure shows the results of the fit for a very low SNR case, limited to ramps without CR hits to disentangle the intrinsic effects of the different likelihood definitions from possible extrinsic effects due to false positive or false negative CR detections.

Several facts are worth discussing. First, the **calwf3** histogram extends to negative values of the flux. This is obviously an absurd result. No source can "absorb" photons from the detector, the true flux is always non-negative. Therefore the individual **calwf3** results must be regarded as possibly wrong. However the mean value of the flux over 10,000 realizations of the same ramp is correct. On the contrary, our approach always returns a non negative value. This behavior is built into our own definition of the likelihood and our explicit distinction between the true flux, enclosed in the x_i latent variables, and the observed flux, represented by the measured y_i's. The output histogram from our `IterativeFitter` runs has a longer tail at large values, although the lack of negative results makes the overall variance of the results for the two methods very similar (see the first row of Table 1, NO CR case).

A way of explaining the difference in behavior between the two pipelines is that, in general, there is no guarantee that a maximum likelihood estimator should be unbiased. However, **calwf3** generalized least squares approach is in fact equivalent to finding the mean of a normal distribution in a maximum likelihood sense. The maximum likelihood estimator of the mean of a normal distribution is unbiased, thus **calwf3** results are unbiased. Nevertheless the Gaussian approximation for the likelihood, as we have amply shown, does not truly apply. Photo-electrons follow a Poisson distribution and the two differ greatly at low values of the Poisson mean. This results in the possibility of finding absurd negative results as well. Our likelihood has a complex form, resulting from the product of multiple Poisson distributions (in the general case in which the time intervals are of different lengths) and multiple Gaussians. For a generic distribution, maximum likelihood estimators may be biased, although, given that we have properly described the likelihood for our ramp, we never observe absurd outcomes.

It is worth noting that in Table 2 for the low SNR case (flux equal to 0.04 electrons per second) the bias in our maximum likelihood estimate decreases with decreasing number of samples. This can be appreciated in better detail by looking at the NO CR columns of the table. This results might seem counterintuitive, but we have a a possible interpretation: as the number of intervals decreases, the length of each interval increases. This in turn means that given the constant input flux, the number of photo-electrons in each interval increases on average as well. This in turn makes the Poisson component of the likelihood more similar to a Gaussian, thus bringing us closer to the regime where the maximum likelihood estimator is unbiased.

10.4 CR identification

In Section 10.1 we noted that our approach is much more efficient at flagging CR hits than **calwf3**. The `IterativeFitter` typically identifies CRs up to much lower energies than **calwf3**, as visible in Figure 6. However, we also mentioned that our method has a higher rate of false positives. To decrease the number of false positives we run our `IterativeFitter` on a subset of the ramps listed in Table 1, while using a higher threshold for CR detection, $\delta = 6$, instead of the default 4. Results are shown in Table 3.

The adoption of a higher threshold completely curbs down the number of false positives, allowing us to reach a purity similar or better than **calwf3**. At the same time, the false negatives are somewhat increased with respect to the $\delta = 4$ case, a predictable outcome. Nevertheless the false negatives rate of our method is still much lower than the **calwf3** one. Another outcome of adopting a larger threshold is that for ramps affected by CR hits (YES CR ramps cases in the Table), the larger number of false negatives leads to both a slightly more biased mean value and a slightly higher standard deviation, again an expected outcome of underdecting CR hits with respect to the $\delta = 4$ case.

It is worth noting that the number of false positives in the $\delta = 4$ case is much higher than a naive expectation from noise and σ-threshold considerations. In a pure Gaussian case, the expected frequency of outliers at a 4σ level is 1 in $\sim 15,000$. With our 10,000 realization and 12 intervals (in the examples of Table 3 we are dealing with an NSAMP=12 type of WFC3/IR ramp) we expect about 8 false positives. We therefore investigate why we observe more false positives than expected. We focus on the second row of Table 3, where

Inputs		Fitter	CR outputs			All ramps		No CR ramps		Yes CR ramps	
Flux	CR#		Det	FP	FN	μ	σ	μ	σ	μ	σ
0.025	4959	This work(CRthr=4)	4739	106	220	0.039	0.034	0.026	0.015	0.059	0.044
		This work(CRthr=6)	4625	0	334	0.053	0.044	0.037	0.023	0.079	0.055
		calwf3	2852	5	2107	0.110	0.256	0.025	0.014	0.239	0.372
0.100	5060	This work(CRthr=4)	4877	41	183	0.094	0.035	0.089	0.022	0.102	0.046
		This work(CRthr=6)	4788	0	272	0.100	0.040	0.094	0.026	0.110	0.052
		calwf3	3001	6	2059	0.185	0.302	0.100	0.017	0.316	0.448
2.500	4871	This work(CRthr=4)	4610	42	261	2.498	0.066	2.496	0.058	2.501	0.078
		This work(CRthr=6)	4482	3	389	2.501	0.069	2.497	0.053	2.507	0.089
		calwf3	2805	4	2066	2.587	0.281	2.500	0.053	2.722	0.409
25.000	4999	This work(CRthr=4)	4361	30	638	25.012	0.189	24.998	0.159	25.034	0.225
		This work(CRthr=6)	4052	6	947	25.025	0.201	24.999	0.159	25.065	0.248
		calwf3	2619	5	2380	25.123	0.397	24.999	0.160	25.318	0.551

Table 3: Fit results for a subset of the tests of Table 1 with extra `IterativeFitter` runs that use a CR detection threshold $\delta = 6$, compared to the standard case of $\delta = 4$ as well as to the **calwf3** corrresponding results.

the input flux of 0.1 results in a typical SNR of 8.5, thus far enough from the problematic regime described in Section 10.3, allowing us to isolate the CR false positive problem from the possible biases in estimating the true flux.

A detailed account of all the false positive cases for $\delta = 4$, reveals that of the reported 41, only 31 unique ramps are affected, i.e. 10 false positives correspond to ramps where more than one CR was falsely identified. Of these 31, only 10 do not have a real CR hit in the input ramp. These 10 are the *true* false positives, where no input CR exists and 1 CR is falsely detected. This is the number that should be compared to the expected occurrence of 8, and, in fact, the two are very close, giving us confidence that the algorithm is performing as expected.

Of the other 21 ramps with false positives, 12 have a real CR hit, and 9 have 2 or more CR hits in input. What happens in these cases is that either the real CR hit is missed and the true flux is mis-estimated, or, even if identified, the CR estimated energies are wrong enough that the true flux is wrongly estimated in this case as well. If the true flux is wrongly estimated, there can be intervals that in practice have observed fluxes that are inconsistent with such erroneous estimates and are thus flagged as CR hits.

It is worth mentioning that our pipeline flags CR hits when the absolute deviation from the estimated true flux is larger than δ times the estimated error for that interval. Many of the CR false positives have negative residuals rather than positive ones. This happens when CR hits are not flagged correctly, and the resulting estimated flux is overestimated, leading to some interval having a flux that is too low with respect to such erroneous overestimate. A possible mitigation strategy would be to flag CR hits only when a positive excess is seen,

rather than using the absolute value of the deviations.

In summary, although our detection algorithm is not perfect, we believe we fully understand its behavior. We correctly predict the CR deposited energy detection threshold, as well as the expected rate of false positives, and we are aware that under some circumstances low-energy single hits or multiple hits can lead to wrong flux estimates, sufficiently different from the truth that they can cause false positive CR detections. The overall performance of the `IterativeFitter` class in flagging CRs is better, and more easily characterized, than what currently adopted in **calwf3**. Increasing the detection threshold can easily rid the user of most of the false positive detections, at a minor cost in deposited energy detection threshold.

10.5 Variable background

We also run simulations for a series of variable background cases, ranging both in average background intensity and background variability (quantified as the root-mean-square). These tests are intended to verify what happens when the underlying assumption of a constant incident flux fails. Brammer (2016) showed that **calwf3** can be tricked into mis-identifying a variable background for CR hits. Depending on the details of which reads are flagged, this may in turn result in over- or under-estimates of the true time-averaged flux. This is visible in **calwf3** images as enhanced-variance and multimodal distribution of the background noise.

We use the same SPAR100, NSAMP=14 read sequence of Brammer (2016) to conduct our tests. Figure 8 shows the outcome. For all these tests we do not add CR hits, in order to simplify the interpretation.

The first row of the Figure shows a baseline test without any added background, with a true flux of 0.5 electrons per second. We are in a regime of moderate SNR of about 25. The overall outputs of our method and **calwf3** are very consistent. The central column of the Figure shows the distribution of p-values obtained for our fits, derived as described in Section 7.6. The baseline case shows a pronounced peak of good fits at p-value of 1, and a long tail of values all the way to p-value of 0. Note however that the p-value plots are in logarithmic scale.

In all the other rows of the figure we show what happens when a background with time-averaged flux of 1 electron/s is added. Between the 2nd and 4th row we tune the amount of variance of the time-dependent background signal. This is quantified by the ratio of background rms divided by the background average. Going from top to bottom the background is progressively more variable. In the second and third row we can see that both our method and **calwf3** recover the average flux of $0.5 + 1 = 1.5$ electrons per second extremely well. Without any further insight we would ot be able to say anything about these fit outcomes. However our definition of a goodness of fit test allows is to appreciate that going from top to bottom the quality of fit is worsening. For a slowly varying, almost constant background of the second row, the p-value distribution starts to become different from the baseline case. In the third row the difference is macroscopic. Although individual p-values still span the full 0-to-1 range, and thus we cannot use individual p-value estimates to flag individual results, it is clear that the p-value distribution when studied over the ensemble of 10,000 simulated ramps, suggests that something is off, although the average flux is correctly retrieved. The extra information carried by the goodness of fitness test and

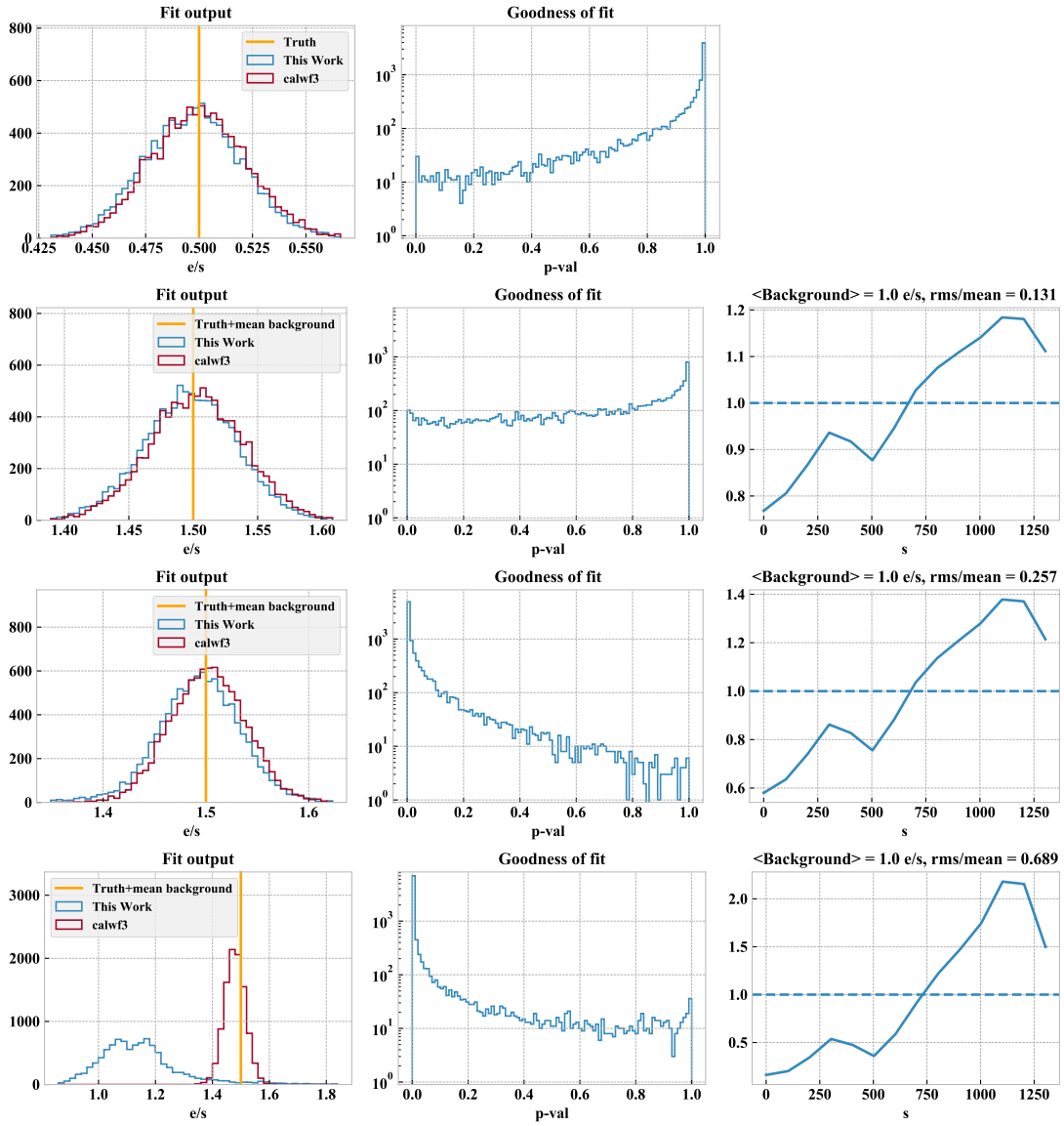

Figure 8: Fit results for different levels of variable background. The "true" input flux is always 0.5 e/s, however in the second to fourth row a varaible background is added, always with mean of 1 e/s. Although the mean is the same, the added background is more variable (in the root-mean-square sense) going from top to bottom. **Left Column:** histogram of the fit results. **Middle column:** distribution of the p-value for the fitted ramps. **Right column:** the time varaible background. **First row:** baseline behaviour in absence of variable signals. **Second to Fourth row:** fit results in presence of varaible background

the p-value distribution allows us to flag the ensemble of the results as being inconsistent with the constant flux null hypothesis.

The fourth row of the figure shows that our pipeline, which is more sensitive to CRs than **calwf3** flags the highly time-variable background as CRs. The number of false positives is extremely high, greater than 30,000, meaning that more than 3 intervals within each of the

10,000 simulated cases are flagged as hit by CR on average. Depending on which intervals are flagged, the output results have a different true flux estimate, resulting in the broad, multimodal distribution of the left panel, bottom row in Figure 8. Once more the p-value distribution allows the user to realize that something must be wrong with the underlying assumption of constant flux.

This p-value approach could be modified to be adopted by **calwf3** by introducing a traditional χ^2 goodness of fit test in the pipeline. The χ^2 statistics could be defined simply as squared sum of the residuals:

$$\chi^2 = \sum_{i=1}^{N_{samp}} \left(\frac{O_i - E_i}{\sigma_i} \right)^2 \tag{35}$$

where O_i are the observed counts between reads, and E_i the expected ones, while σ_i is the estimated error on the counts for that interval, incldluing the readnoise. Intervals with identified CRs should be skipped. The distrubution of the resulting p-values could be used to assess whether the underlying hypothesis of constant flux holds over the ensemble of tests.

11 The JWST case

As we have previously shown, in particular in Section 3, our method has been developed to be able to account for groups with multiple averaged and skipped frames, and can be thus applied to *JWST*-like read sequences. Robberto (2014) has developed a generalized least squares (GLS) approach to fitting near-IR non-destructive ramps with averaged and skipped frames. At the time of publication the work argued that the weighted least squares (WLS) estimator, initially baselined for the *JWST* data reduction pipeline suffered of sever shortcomings when compared to a GLS approach. WLS is a generalization of the ordinary least squares (OLS) approach, in which knowledge of the variance of observations is incorporated. The generalized least squares approach offers an even more complete approach to treating the uncertainties, as it allows to introduce a full variance-covariance matrix for the averaged groups. When correctly propagated into the solution for the flux, this allows a better estimate of its variance. Although the OLS/WLS approach for estimating the mean of the true flux is itself unbiased (as the GLS one), Robberto (2014) showed that OLS/WLS may severely underestimate the uncertainties. The work by Robberto (2014) was instrumental in pushing for including a GLS option in the *JWST* pipeline using the variance-covariance definitions derived by Robberto (2014) and the previous publications of the same series (Robberto, 2009a,b,c). Note however that by default the JWST pipeline will be using a WLS approach with weights based on the optimal weighting scheme by Fixsen et al. (2000)[1].

We use the `IterativeFitter` class to perform a subset of the experiments in Robberto (2014). Our results are shown in Table, together with the Robberto (2014) ones. The agreement with the GLS approach of Robberto (2014) is, not surprisingly, almost perfect. This is expected because the ramps in these examples are simulated for a very high SNR regime,

[1]See `https://jwst-pipeline.readthedocs.io/en/latest/jwst/ramp_fitting/description.html#slope-and-variance-calculations`

	This work		Robberto (2014)				
Case	μ	σ	μ_{GLS}	σ_{GLS}	μ_{OLS}	σ_{OLS}	σ_{theory}
m=1	8.9882	0.3788	8.9967	0.3666	8.9967	0.1793	0.3707
m=4	8.9987	0.1605	9.0002	0.1539	8.9993	0.0413	0.1660

Table 4: Comparison of our approach and the results of Robberto (2014). GLS stands for Generalized Least Squares, the approach proposed by Robberto (2014), while OLS stands for Ordinary Least Squares. The theoretical estimation of the standard deviation, σ_{theory} is also derived in Robberto (2014). In both cases the true flux is 9 electrons per second, the gain is 1 and the σ_{ron} is 15 electrons. The frame time is 10 seconds and 10 groups are used. In one case only 1 frame per group is used, while in the other 4 frames are averaged together.

where the Poisson terms in our likelihood are effectively very similar to normal distributions. One of the GLS assumption, clearly spelled out by Robberto (2014) is indeed that the underlying noise model is Gaussian, similarly to the approach of **calwf3**. Differences, if any, would be observed at low SNR but should not, and are not, observed at high SNR. Nevertheless, this very good agreement corroborates both ours and the Robberto (2014) approaches, and in particular validates the complicated derivations of the variance-covaraince matrix terms. The latter, evidently are correct in both works (or equally wrong!). This gives further confidence that the right choice for the ramp fitting in the *JWST* pipeline is indeed a GLS approach and not an OLS one.

12 Summary and Conclusions

We described the full likelihood for non-destructive ramp measurements in IR detectors. We derived an iterative method for finding the maximum of such likelihood as a function of the incident flux, while efficiently identifying CR hits. Our method can be applied to the case of multiple averaged and skipped frames per group.

We compared our method to the **calwf3** pipeline for the WFC3/IR detector on the Hubble Space Telescope, as well as to the generalized least squares fitting method derived by Robberto (2014) and currently implemented in the *JWST* pipeline (although the default version of the JWST pipeline will use optimally weighted, but not generalized least squares). Both these methods are based on least-squares minimization, that is equivalent to maximizing a Gaussian likelihood. The results show good agreement in the high SNR regime and several differences in the low SNR regime. In particular, least-squares methods are very good at producing unbiased results, although giving at the same time absurd, negative values for the incident flux. Our method provides a slightly biased estimate for very low fluxes, but never an unphysical one. These differences are ascribed to our method using a full likelihood which correctly describes the incident flux as a Poisson variable, opposite to least-squares methods that effectively approximate the flux as a normal variable. This approximation is valid in high SNR regime, but fails at low SNR. Our detailed results demonstrate that special caution must be exercised in interpreting the results in the regime of $f \to 0$.

We also demonstrate that our algorithm is far superior to the one currently used in **calwf3**

for identifying CR hits. Our detection rate is much higher, the exact values depending on the detailed characteristics of the ramps time sequences and incident fluxes. Moreover our algorithm provides a clearly quantifiable way to exercise trade-offs between CRs deposited energy detection threshold versus the rate of false positive and false negatives. The JWST pipeline will be using a CR detection algorithm similar to ours, and not based on the **calwf3** one.

Finally, we showed how to test the hypothesis that the underlying flux is truly constant by introducing a goodness of fit measure in the form of a p-value. This can be used to identify cases in which an ensemble of ramps (e.g. all the pixels in an image) is adversely affected by time-variable background.

We conclude by recommending that modifications to the **calwf3** pipeline and specifically the `CRCORR` step CR identification algorithm be considered, switching to algorithms along the lines of the algorithm devised within our `IterativeFitter` class. Our approach basically consists in identifying outliers in the differences between successive reads and is also the approach that the *JWST* pipeline will be using.

We further recommend to evaluate and consider the introduction of an extension in the WFC3/IR .flt fits files produced by **calwf3** (or equivalently a new, additional data product) to contain a measure of the goodness of fit for each ramp (i.e. for each pixel). Although our goodness of fit test cannot be directly ported to **calwf3**, a simple χ^2 test could serve the purpose of alerting the user that something is amiss with their data, in case of the presence of a variable signal. A p-value would allow the user to flag pixels affected by variable background, but could also be used to identify other transients, e.g. moving objects streaks, artificial light from terrestrial sources in the case of WFC3/IR ground flats, and more.

All of the code used in this work is provided and documented on `https://github.com/spacetelescope/Up_the_ramp`. The repository includes several example notebooks to help the user getting started.

Acknowledgements

M. Gennaro would like to thank Dr. Kirill Tchernyshyov for the initial discussion that inspired this work and Dr. Massimo Robberto for the never ending discussions on all the topics touched by this paper: noise, detectors, statistics, software, career, music and more.

References

Barker, E. A., McCullough, P., & Martel, A. R. 2009, WFC3 IR SAA Passage Behavior, Space Telescope WFC Instrument Science Report, ,

Brammer, G. 2016, Reprocessing WFC3/IR Exposures Affected by Time-Variable Backgrounds, Tech. rep.

Dahlen, T., McLaughlin, H., Laidler, V., et al. 2008, Improvements to Calnica, Tech. rep.

Dressel, L. 2019, Wide Field Camera 3 Instrument Handbook, Version 12.0 (Baltimore: STScI)

Fixsen, D. J., Offenberg, J. D., Hanisch, R. J., et al. 2000, PASP, 112, 1350

Nelder, J. A., & Mead, R. 1965, The Computer Journal, 7, 308. https://doi.org/10.1093/comjnl/7.4.308

Rauscher, B. J., Fox, O., Ferruit, P., et al. 2007, PASP, 119, 768

Regan, M. 2007, JWST-STScI-001212

Robberto, M. 2009a, JWST-STScI-001853

—. 2009b, JWST-STScI-002128

—. 2009c, JWST-STScI-002161

Robberto, M. 2014, in Proc. SPIE, Vol. 9143, Space Telescopes and Instrumentation 2014: Optical, Infrared, and Millimeter Wave, 91433Z

A Code Examples

A.1 Using the `RampTimeSeq` class

The examples in Listings 2 and 3 illustrate how to initialize and use this class and how to plot its output using its convenience plotting method.

For example the code:

```
import numpy as np
from ramp_utils.ramp import RampTimeSeq

dt, nf, ns, ng = 10., 4, 5, 5
myramp = RampTimeSeq('GENERIC', ng, nframes=nf, nskips=ns,
                     read_times=dt*np.arange(ng*(nf+ns)))
myramp.test_plot()
```

Listing 2: An example using the `RampTimeSeq` class to instantiate and plot a `GENERIC` ramp

produces the timing sequence shown in Figure 9,

while the code:

```
import numpy as np
from ramp_utils.ramp import RampTimeSeq

myramp = RampTimeSeq('HST/WFC3/IR', 15, samp_seq='SPARS100')
myramp.test_plot()
```

Listing 3: An example using the `RampTimeSeq` class to instantiate and plot a `WFC3/IR` ramp

produces the timing sequence shown in Figure 10, taking advantage of the predefined WFC3/IR sequences.

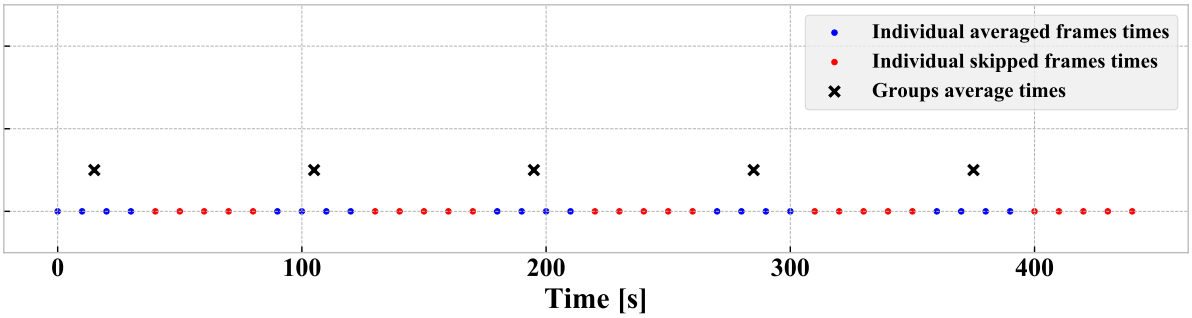

Figure 9: `GENERIC` ramp time sequence generated with the above commands

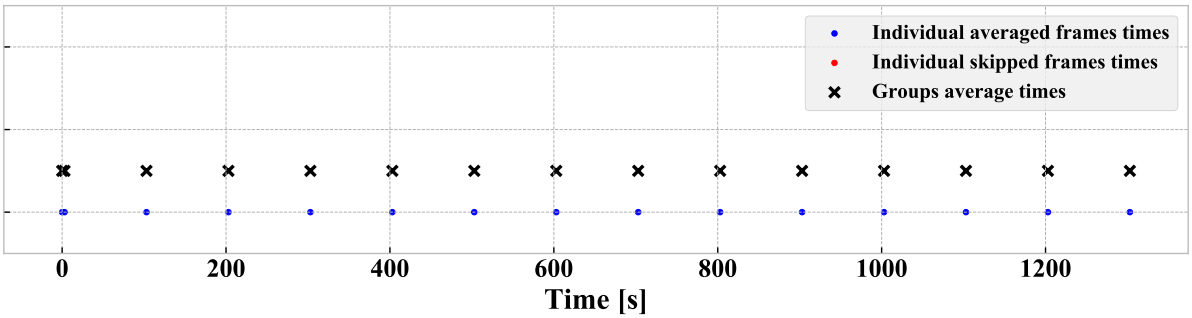

Figure 10: `WFC3/IR, SPARS100` ramp time sequence generated with the above commands. Note the two samples at 0 and 2.9 seconds, almost coincident in the plot due to the horizontal scale. Also note that this sequence has been generated using an input value of 15 for the number of samples, however in WFC3/IR parlance, this is a NSAMP=14 case, given the fact that the zero-th frame is not counted in the official WFC3/IR numbering of read sequences (see Dressel, 2019). In practice the zero-th sample is always subtracted from the subsequent ones in **calwf3**, thus removing the intercept and reducing the linear fitting problem to a slope fitting with fixed intercept, and this process is equivalent to using all 15 samples.

A.2 Using the `RampMeasurement` class

This section illustrates the basic functionalities of the `RampMeasurement` class: how to generate a ramp and how to add CRs. The code in Listing 4 generates the two plots of Figure 11:

```
import numpy as np
from ramp_utils.ramp import RampTimeSeq,RampMeasurement

#First generate a time sequence
dt, nf, ns, ng = 10., 3, 2, 8
myramp = RampTimeSeq('GENERIC', ng, nframes=nf, nskips=ns,
           read_times=dt*np.arange(ng*(nf+ns)))

#Then pass the generated time sequence, the flux value and the other
#required inputs to the RampMeasurement class, and generate a
#measurement, displayed in the left column of Figure 11
```

```
myflux = 1.
mymeas = RampMeasurement(myramp, myflux, gain=1., RON=15., KTC=0,
          bias=10000,full_well=100000)
mymeas.test_plot()

#Generate the same type of measurement as above,
#but add cosmic ray hits as well, as visible in the
#right column of Figure 11
myCRdict = {'times':[115.,290.],'counts':[800,1000]}
mymeas = RampMeasurement(myramp, myflux, gain=1., RON=15., KTC=0,
          bias=10000, full_well=100000, CRdict=myCRdict)
mymeas.test_plot()
```

Listing 4: An example using the `RampMeasurement` class to generate a ramp measurement from a ramp time sequence both with and without cosmic ray hits

We can see how the individual frames, filled circles, are averaged together to give the 'X' measurements and how the various sources of noise are added to the noiseless counts. In the right column, two CR hits are visible as jumps in the counts. A closer look shows that the first CR hits at 115 seconds, i.e. between two averaged frames, while the second CR hits at 290 seconds, i.e. between skipped frames. The effect on the average group values is that in the first case there is a gentle 2-step jump, while in the second case the jump is more abrupt. In the first case both the count differences before and after the hit are effectively lost when recovering the flux, while in the second hit case, only one interval is lost.

A.3 Using the `RampMeasurement.add_background` method

The `add_background` functionality is shown in Listing 5 for a `HST/WFC3/IR` ramp. We first define a variable background trend, and normalize it to the desired average background flux. The background used in this example is shown Figure 12.

We then generate a Poisson, random number of background electrons in each interval. The cumulative number of background electrons is added to our ramp measurement. The code below illustrates how to achieve this with the `RampMeasurement` class, while Figure 13 shows the output of the code, i.e. two ramp measurements that differ because in the second case a variable background was added.

```
import numpy as np
from scipy.interpolate import interp1d
from ramp_utils.ramp import RampTimeSeq,RampMeasurement

# Generate a ramp without background
# (Left column of Figure 13)
myramp = RampTimeSeq('HST/WFC3/IR',15,samp_seq='SPARS100')
myflux = 1.
CRdict = None  # For no cosmic rays
mymeas = RampMeasurement(myramp,myflux,CRdict=CRdict)

# Make up a background trend (in electrons/s)
bg_times      = np.linspace(0,1400,10)
bg_electron_rate = np.array([1.,1.2,1.7,1.3,1.7,2.5,4.2,5.4,4.0,3.5])
```

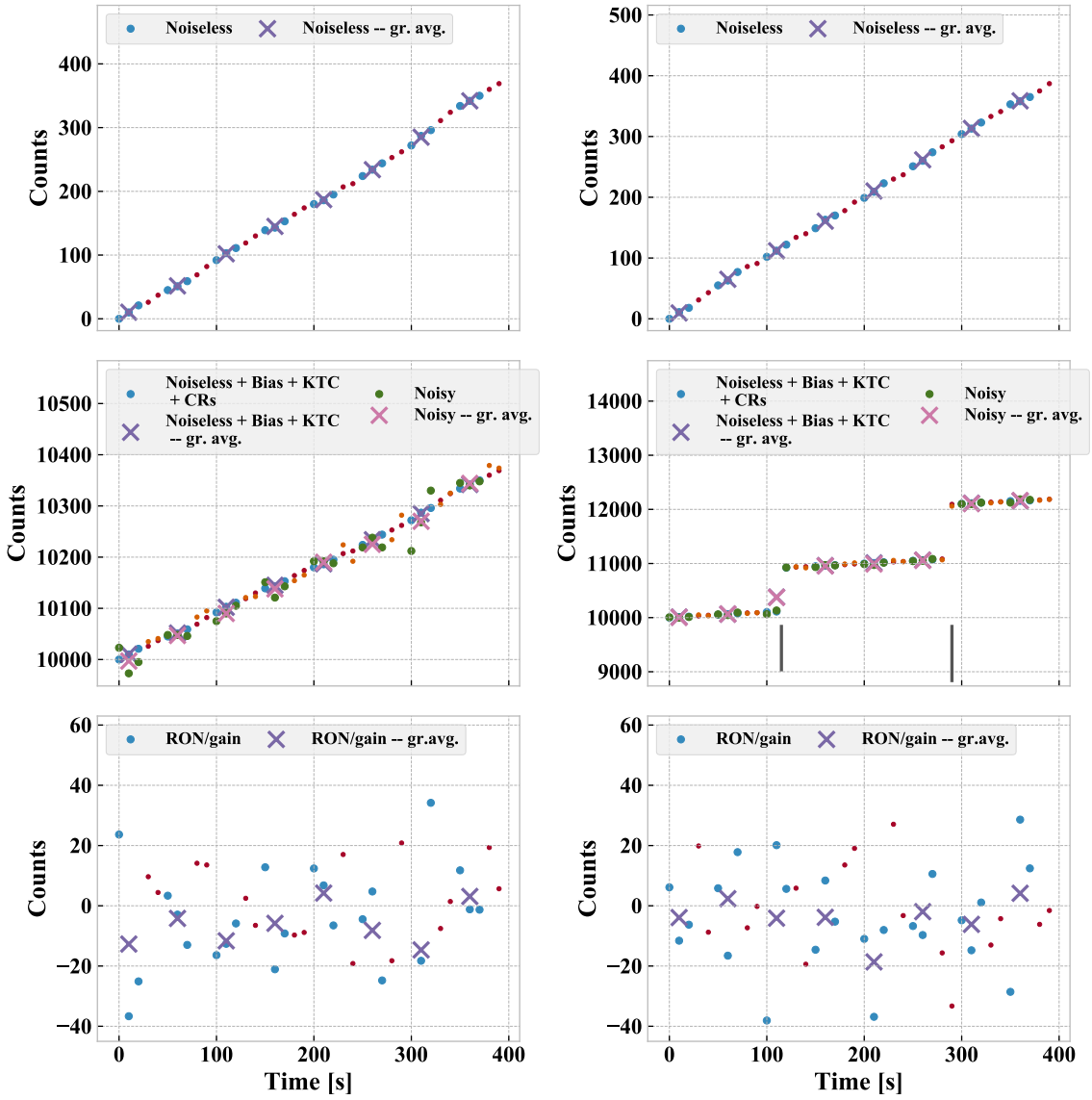

Figure 11: Left: ramp measurements for a ramp with 8 groups and 3 averaged and 2 skipped frames per group. **Right:** similar to the left case, but adding two CR hits. **All panels:** values for individual frames are shown as circles, values for the averaged groups as crosses. Skipped frames are also shown as smaller circles, and are not described in the legend for ease of reading. See Figure 4 for a description of the top-center-bottom panels.

```
# Compute the background at each of the ramp times
bg_int = interp1d(bg_times,bg_electron_rate,'quadratic')
varbg = bg_int(myramp.read_times)

# Compute the time averaged background value
# and normalize at the desired rate
# (see Figure 12)
dt = myramp.read_times[-1]-myramp.read_times[0]
```

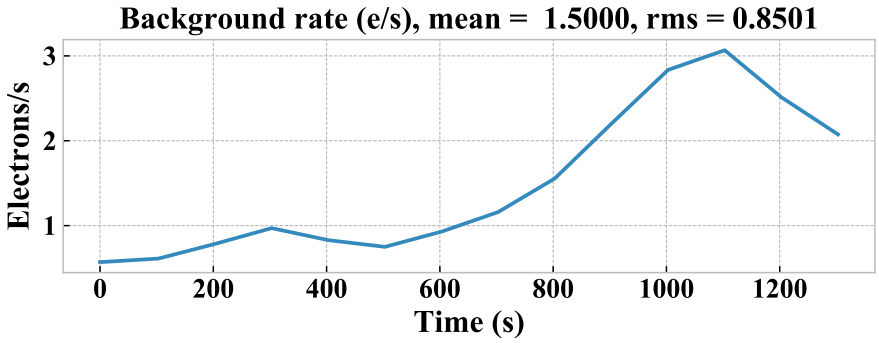

Figure 12: Variable background model

```
t_avg = np.trapz(varbg,myramp.read_times) / dt
mean_bg_electron_rate = 1.5
varbg = varbg/t_avg * mean_bg_electron_rate

# Go from rate to electrons generated within each interval
bg_electrons=[0]
mean_bg = 0.5*(varbg[1:]+varbg[:-1])
dts = mymeas.RTS.read_times[1:]-mymeas.RTS.read_times[:-1]
bg_electrons.extend([np.random.Poisson(lam=vb*dt)
         for vb,dt in zip(mean_bg,dts) ])
bg_electrons = np.asarray(bg_electrons)

#Pass the cumulative value to the add_background
# method of the RampMeasurement object
#(Right column of Figure 13)
mymeas2 = copy.copy(mymeas)
mymeas2.add_background(np.cumsum(bg_electrons))
```

Listing 5: An example using the `add_background` functionality to add a variable background to a ramp measurement

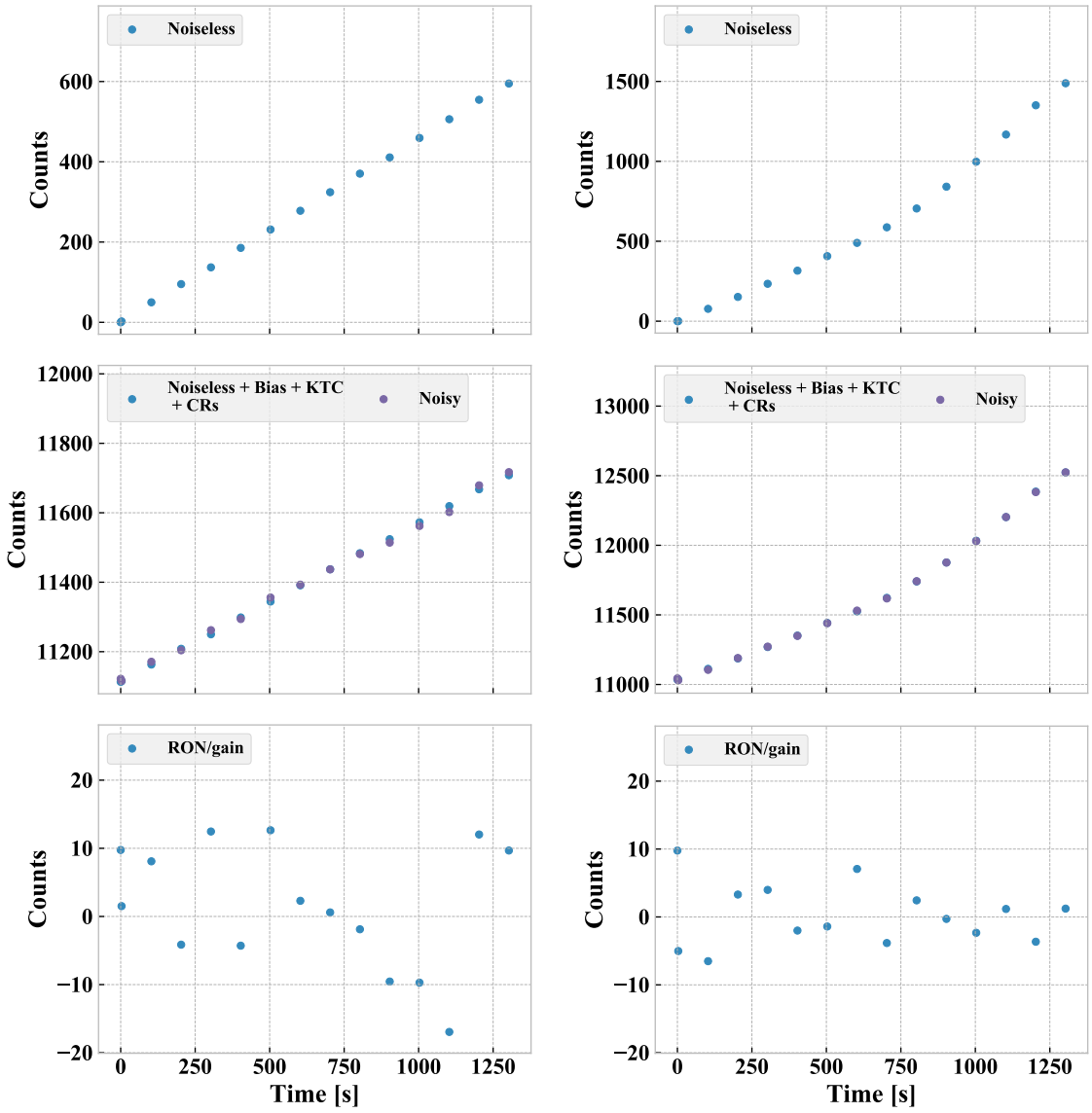

Figure 13: Left: an example of a `RampMeasurement` object for a WFC3/IR SPARS100 read sequence. **Right:** the same but with added a variable background flux. See Figure 11 for a description of the top-center-bottom panels.