Draft version March 27, 2023
Typeset using LaTeX **preprint2** style in AASTeX631

DRAFT (Rev. 1.5) NSClean: An Algorithm for Removing Correlated Read Noise from JWST NIRSpec Images

Bernard J. Rauscher[1]

[1] *NASA Goddard Space Flight Center*
Observational Cosmology Laboratory
8800 Greenbelt Road
Greenbelt MD 20771, USA

Cite as Rauscher, B.J. (2023), PASP, in preparation

ABSTRACT

NSClean is an algorithm and associated python package for removing faint vertical banding and "picture frame noise" from JWST Near Infrared Spectrograph (NIRSpec) images. NSClean uses known dark areas to fit a background model to each exposure in Fourier space. When the model is subtracted, it removes nearly all correlated noise. Compared to simpler strategies like subtracting the rolling median, NSClean is more thorough and uniform. NSClean is computationally undemanding, requiring only a few seconds to clean an image on a typical laptop. The NSClean package is freely available for download from the NASA JWST website (NASA JWST website 2023).

1. INTRODUCTION

JWST is today's premier observatory for mid and near-infrared (NIR) space astronomy. To enable science objectives cutting across astrophysics, JWST carries a suite of four science instruments: a Near Infrared Camera (NIRCam; Rieke et al. 2022), a Near Infrared Imager and Slitless Spectrograph (NIRISS; Doyon et al. 2022), a Mid-infrared Instrument (MIRI; Rieke et al. 2022), and a Near Infrared Spectrograph (NIRSpec; Jakobsen et al. 2022). This article concerns NIRSpec, an algorithm and associated software for further reducing its already low noise: "NSClean". NSClean should be beneficial to most NIRSpec Integral Field Unit (IFU) and many Multi-Object Spectrograph (MOS) observers.

From early on, it was understood that NIRSpec required ultra-low noise detectors. It is detector noise limited for all but prism-mode observations. This is in contrast to other JWST instruments that are generally limited by the astronomical background. Consequently, NIRSpec has lower noise requirements than other JWST instruments. "Total noise" is a concept that was introduced for JWST. To measure it; one defines a standard scientific exposure, takes many such exposures (typically >40), and then computes the standard deviation per pixel. Across JWST's NIR instruments, the exposure time was taken to be 1000 seconds. For NIRCam and NIRISS, median total noise was re-

Corresponding author: Bernard J. Rauscher
Bernard.J.Rauscher@nasa.gov

quired to be $<10\ e^-$ per exposure. For NIRSpec, the requirement was $<6\ e^-$.[1]

NIRSpec's $<6\ e^-$ noise requirement is the reason why we developed Improved Reference Sampling and Subtraction (IRS^2; pronounced IRS-square; Rauscher et al. 2017). In IRS^2 mode, NIRSpec uses a special clocking pattern and reference correction pipeline step to reduce correlated noise as far as possible using the NIRSpec detector's built-in references. Using IRS^2, NIRSpec's total noise is slightly $<6\ e^-$ on average, and to within the uncertainties compliant with requirements. IRS^2 is the recommended readout mode for most observations except for extremely bright targets (JWST User Documentation website 2016).

However, even with NIRspec's detectors meeting requirements, many NIRSpec observers report seeing faint, correlated read noise in count rate images that complicates calibration. Fortunately, for NIRSpec, much of this can be removed by using dark areas of images as references.

Figure 1 shows an example of the correlated noise from an early NIRSpec Integral Field Unit (IFU) observation. We have smoothed the images and stretched the greyscales to emphasize correlated noise that would otherwise be more difficult to see against the background of NIRSpec's ∼6 electrons total noise. One sees a "picture frame" effect, whereby areas near he edges of both detectors on all four sides seem less noisy. In the interiors, one sees faint vertical striping. While the amplitude is small, this correlated noise can undermine accurate photometry when no local sky is available. This is often the case for IFU observations and we are aware of cases where this is true also in MOS mode.

[1] MIRI uses a different detector technology for which the comparison is not relevant.

NSClean uses blanked off areas of NIRSpec scenes to model the background, including correlated noise.

Because it uses more information, NSClean's correlated noise correction is more complete and more uniform than is possible without careful masking.

2. PHYSICAL CAUSE OF THE CORRELATED NOISE

Our focus in this paper is on the specific correlated read noise that NSClean is designed to fix. Readers who want to learn more about NIRSpec's read noise in general may want to see some of our earlier papers. Rauscher (2015) describes the origins of NIRSpec's white and $1/f$ noise, and provides a python package for simulating it. Rauscher et al. (2017) describes NIRSpec's IRS^2 readout mode. Without IRS^2, the residual correlated noise that remains today would be much worse.

The correlated noise that remains after IRS^2 is a logical consequence of how IRS^2 works. NIRSpec uses two Teledyne H2RG NIR detector arrays (Loose et al. 2003). Each H2RG provides two types of reference information that can be used to remove correlated read noise. These are the "reference pixels" that form a 4-pixel wide frame on all sides of NIRSpec images and one "reference output" per H2RG. The reference output is not visible in the usual pipeline data products, but it is used most of the time. As described in Rauscher et al. (2017), IRS^2 is built on principal component analysis (PCA) showing that NIRSpec's read noise is covariance stationary to a high degree of approximation. Informally, this means that the read noise is independent of when one looks.

It turns out that in JWST's NIR detector systems, thermal instability causes noise that is not covariance stationary. There is a picture frame pattern that changes in time at the $\sim 1\ e^-$ level. Rauscher et al. (2013) describe how small temperature fluctuations can drive the picture

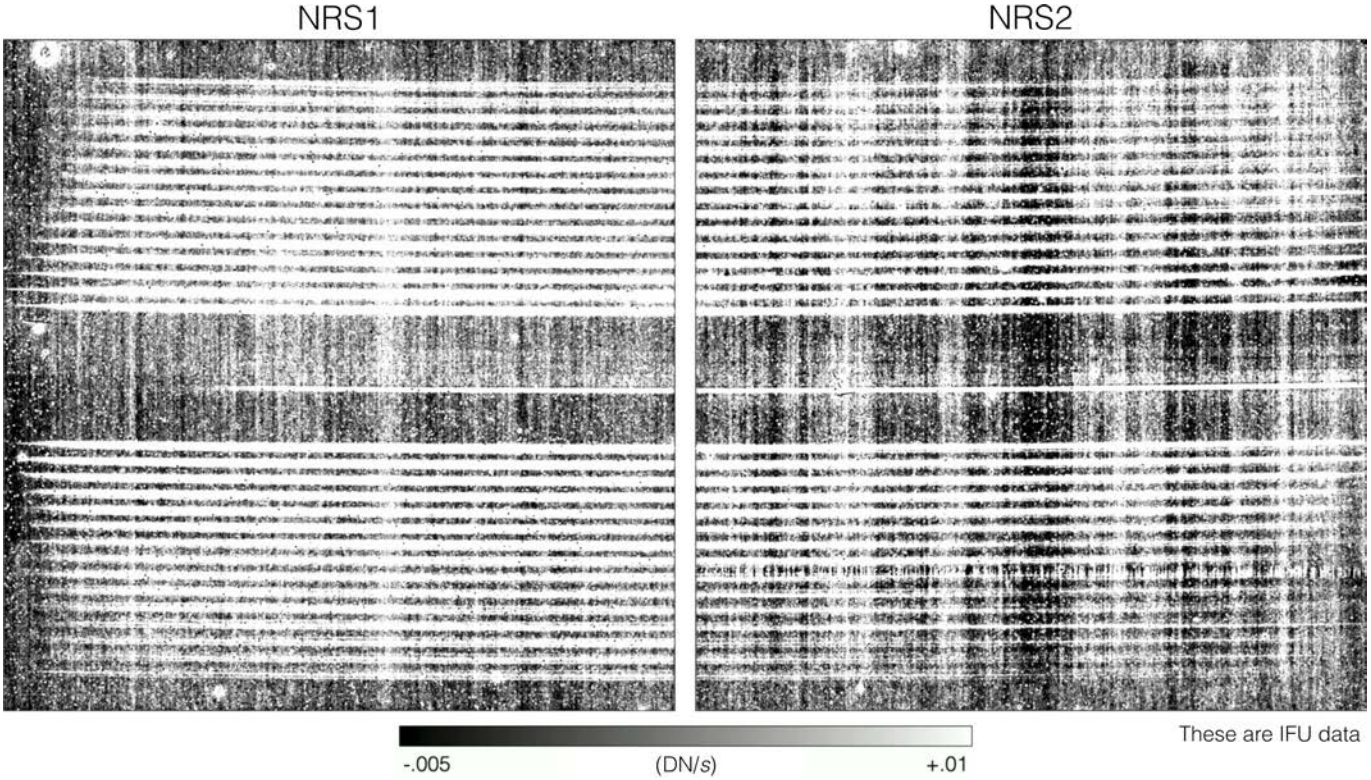


Figure 1. The JWST pipeline makes NIRSpec count rate images like those shown here. This observation used NIRSpec's IFU mode which produces 30 horizontal spectral traces per detector. To highlight correlated noise, we have smoothed the images and set the greyscale roughly equal to NIRSpec's 6 electrons total noise requirement. One sees vertical banding in the central regions of both detectors. Toward the edges of both detectors, there seems to be less correlated noise. This is the "picture frame". While both types of residual noise are less than NIRSpec's total noise requirement, they nevertheless complicate calibration. For example, they can produce negative fluxes and features that mimic emission lines or continuum. NSClean fits a background model to dark areas of each exposure and subtracts it to remove picture frame noise and vertical banding.

frame. This is why the vertical banding that is visible in Figure 3a seems to fade away near the edges. The relatively quiet edges are in the picture frame while the vertical bands are not. IRS^2 relies on the reference pixels to see noise in order to remove it. Since the reference pixels are in the picture frame and do not see the vertical banding, IRS^2 is powerless to remove it.

3. ALGORITHM

NSClean is built on the Fourier transform of the instrumental background. Our treatment starts in Section 3.1, by reviewing how python's numpy package implements the classical Fast Fourier Transform (FFT; Cooley & Tukey 1965) for fully sampled data. Since NIRSpec's background is not fully sampled (because of astronomical sources), Section 3.2 explains how NSClean computes a statistically optimal approximation to the Fourier transform using all available background samples.

The next two subsections describe the linear algebra that underpins NSClean. Insofar as possible, we have tried to use a consistent, standard notation. Throughout this paper, boldface lowercase letters are vectors and uppercase boldface letters are matrices. When discussing matrix elements, we use superscripts for row indices and subscripts for column indices.

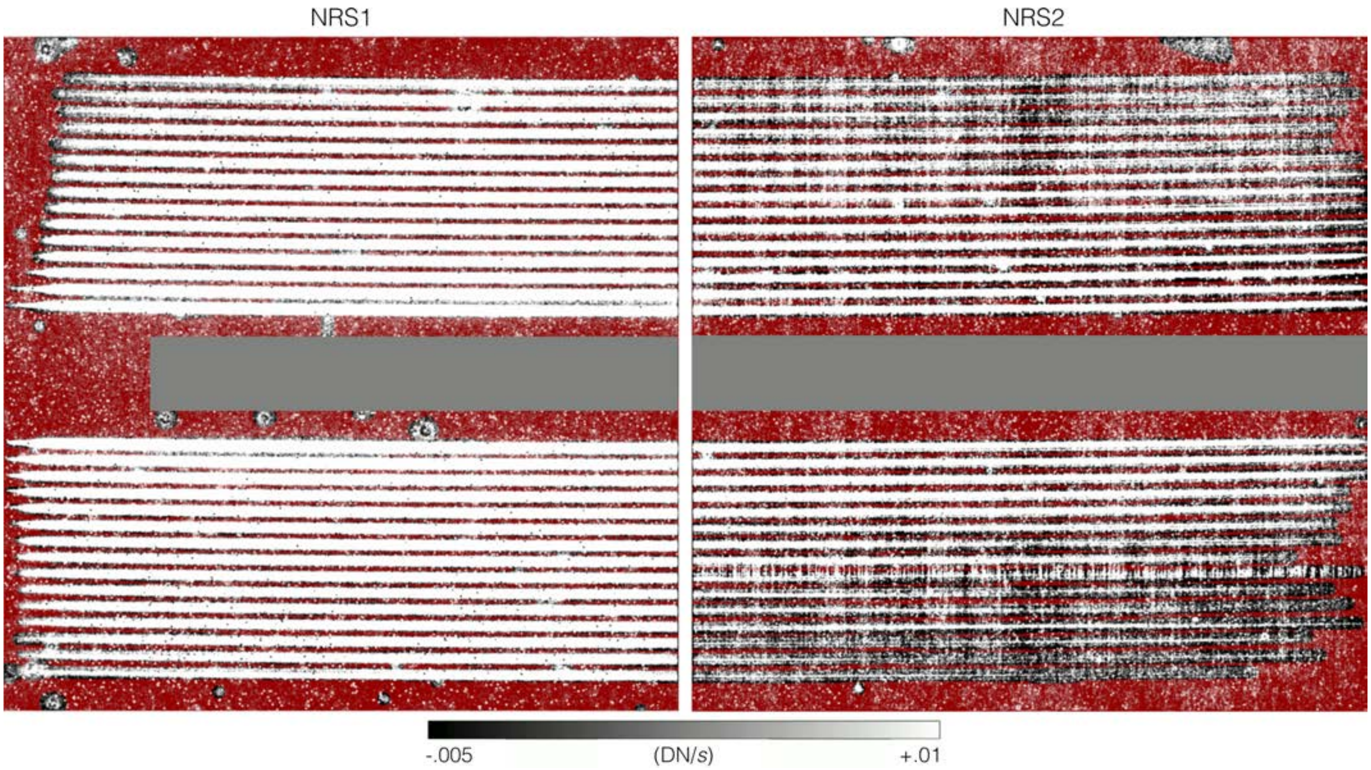


Figure 2. We used the background masks shown here for development. The underlying grayscale image is the median of a stack of illuminated IFU exposures. The 30 spectral traces per detector are clearly visible. We used the red-shaded pixels to make the background model. As described in the text, we used the GNU Image Manipulation Program (GIMP) to manually make the masks since we only needed one set to write the software. We understand that some JWST observers have already automated mask generation. They grey rectangles blank off areas of potentially illuminated (by scattered light) areas of the focal plane that we left unconstrained during background fitting.

3.1. *Numpy's Classical FFT*

For dark exposures, one can use numpy's FFT package to compute the Fourier transform of an image column. Like all FFTs, numpy uses a highly-efficient factorization of the Fourier matrix, $\mathbf{F}$, to solve the matrix equation,

$$\mathbf{Ff} = \mathbf{d}, \tag{1}$$

where $\mathbf{f}$ is the Fourier transform of the data, $\mathbf{d}$. For n pixels per column, in numpy the elements of $\mathbf{F}$ are,

$$F_k^m = \exp\left\{2\pi i \frac{mk}{n}\right\}, \tag{2}$$

where m is the row index and k is the column index. Because NIRSpec's data are real valued and $n = 2048$ is an even number; $m = 0, 1, \ldots, n-1$ and $k = 0, 1, \ldots, n/2$.

3.2. *NSClean's Fourier Transform*

For NIRSpec's incompletely sampled background, NSClean uses weighted least squares to approximate Fourier transforms. The starting point is again equation 1,

$$\mathbf{Ff} \approx \mathbf{d}, \tag{3}$$

but now as an approximation and with the understanding that $\mathbf{F}$, $\mathbf{f}$, and $\mathbf{d}$ are incomplete. $\mathbf{F}$ is missing columns where light falls on the detector and rows for frequencies that we choose not to fit. $\mathbf{f}$ contains only a few very low frequencies to minimize noise. $\mathbf{d}$ is missing rows where the detector is illuminated.

To solve equation 3 using least squares, we minimize the generalized distance squared,

$$\delta^2 = (\mathbf{Ff} - \mathbf{d})^{\mathrm{H}}\mathbf{W}(\mathbf{Ff} - \mathbf{d}), \tag{4}$$

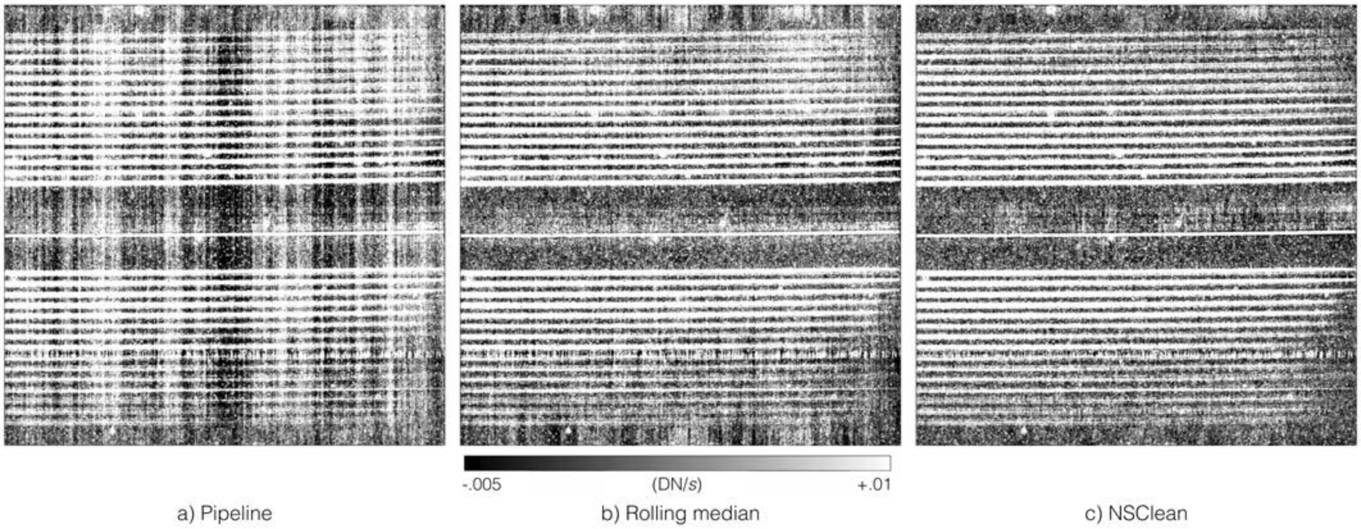


Figure 3. This figure shows the a) correlated noise that is visible in pipeline calibrated images. The actual pipeline products do not look this bad. We have adjusted the grayscale and blurred the images slightly to highlight correlated read noise. Panel b) shows the effect of subtracting the median of a few neighboring columns from each column. The NIRSpec Instrument Team prevously provided a tool to NIRSpec observers that does this. Finally, panel c) shows the NSClean result. Panels b and c are noticeably cleaner than panel a. Comparing panels b and c, panel c shows more uniform and complete background subtraction.

using all available background samples. The symbol, "H", denotes the conjugate transpose, which is also known as the Hermitian transpose. A weight matrix, $\mathbf{W}$, is required to compensate for non-uniform background sampling. The current version of NSClean weights inversely by the local sample density squared, ρ^{-2}:

$$\mathbf{W} = \begin{bmatrix} \rho_{00}^{-2} & 0 & 0 & 0 \\ 0 & \rho_{11}^{-2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \rho_{n'-1\ \ n'-1}^{-2} \end{bmatrix}. \tag{5}$$

$\mathbf{W}$ is diagonal and equal to it's conjugate transpose. Section 3.3 describes $\mathbf{W}$ in more detail. The quantity $n' \leq n$ is equal to the number of background samples. Under these conditions, the least squares solution to Equation 4 is,

$$\mathbf{f} = \left(\mathbf{W}^{1/2}\mathbf{F}\right)^{+} \mathbf{W}^{1/2}\mathbf{d}. \tag{6}$$

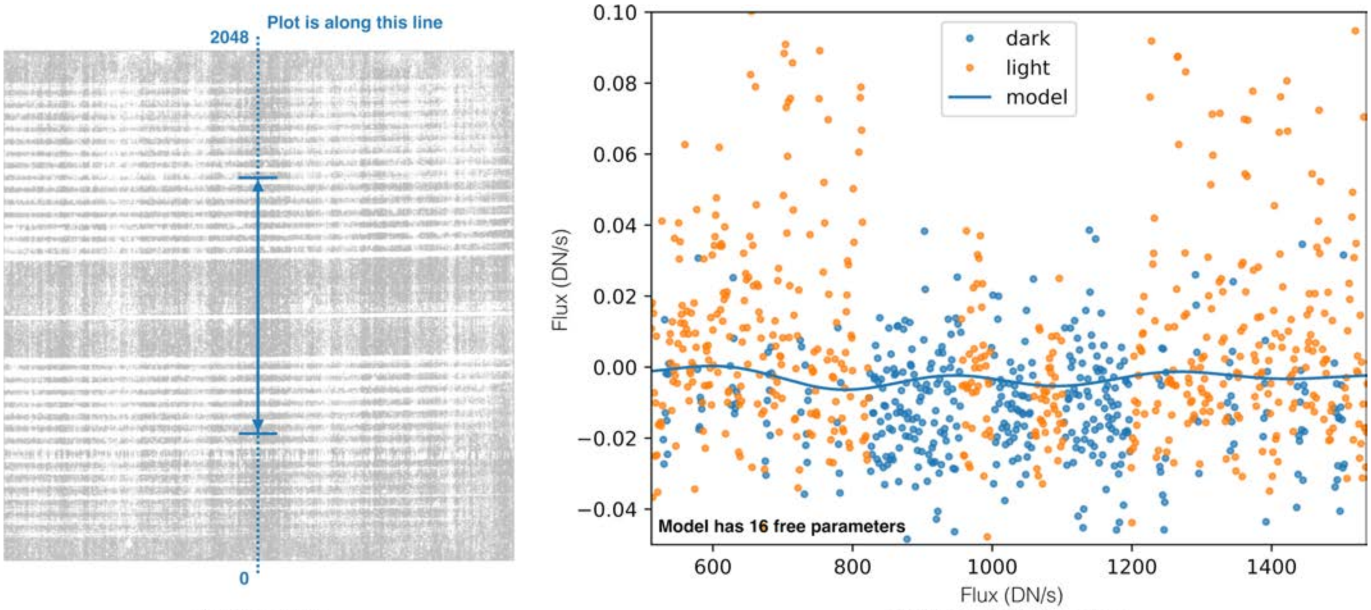
The symbol, "+", denotes the Moore-Penrose inverse. Being a Fourier transform, the quantity $\mathbf{f}$ is a complex valued vector.

Equation 6 is this section's key result. NSClean uses this expression to approximate the Fourier transform of the incompletely sampled background.

Figure 4 shows an example of how equation 6 works in practice. Panel a) shows a vertical cut through NRS2, which is the most affected of the two detectors. To show detail, Panel b) shows only the innermost 1024 rows. The blue points are background samples, the orange points are pixels that the background mask marked as potentially illuminated, and the blue line is the model built using equation 6. As a practical matter, we were able to fit about nine frequencies ($\approx$ 16 free parameters) before we started to see increased noise due to over fitting. As expected, the blue line passes near the centers of groups of blue points. It is smooth, continuous, and very low noise compared to the pixels themselves.

3.3. *The Weight Matrix,* $\mathbf{W}$

The weight matrix compensates for uneven background sampling. Returning to Figure 2,



a) Cut location

a) Model overlaid on data

Figure 4.

there are often only a few rows of blanked off background pixels between the spectral traces. But; near the bottom, middle, and top of each detector, there are much larger areas of background pixels. When nothing is done to compensate for the uneven background sampling, scientifically uninteresting areas of the scene carry far too much weight.

As described earlier, NSClean computes the Fourier transforms of columns individually using weighted least squares fits. After a bit of trial and error, we found that weighting inversely by the local background sample density in columns worked well. There is nothing fundamental about this weighting scheme. We imagine that some observers will find better ones for their data.

One could compute the local sample density by convolving the background mask with a tophat function (Figure 5). While effective, the resulting weight curve is quantized in units of the tophat's width. To eliminate the quantization while still approximating the local density, NSClean convolves columns of the background mask with a Gaussian kernel. In the current release, the kernel's standard deviation is hard coded to be $\sigma = 32$ pixels. Going forward, it may be possible to come up with something more elegant. 32 pixels seems to work well for many IFU observations.

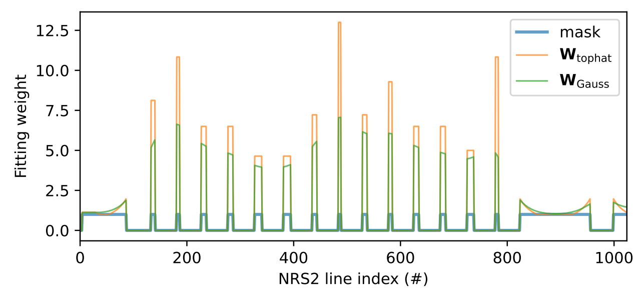


Figure 5. This figure shows the background mask and diagonal of $\mathbf{W}^{1/2}$ along the same vertical cut through NRS2 that is shown in Figure 4a. For clarity, we show only the first 1024 rows. Mask values $=1$ are treated as background and mask values $=0$ are treated as potentially illuminated. The orange curve shows the weights that result from convolving a 65 pixel wide tophat. The green curve shows that weights derived from convolving a Gaussian kernel with $\sigma = 32$ pixels. As described in the text, NSClean uses Gauss-convolution because the resulting weights are more uniform and the weight curve is not quantized.

3.4. *Making Masks*

This section describes how we made the masks that are shown in Figure TBD.

4. IMPLEMENTATION

NSClean is written in python-3. We chose python for compatibility with the rest of the JWST pipeline. The current NSClean version is not computationally demanding. Teams XX, YY, and the JWST Early Release Science (ERS) team TEMPLATES (Rigby et al. 2023, in prep.) have tested NSClean on typical scientific workstations and laptops and report that it works well. The typical cleaning time for one 2048×2048 NIRSpec image is a few seconds. This assumes that multithreading is turned on for the python linear algebra libraries as described in Section 4.2.

The current NSClean version works column-by-column. Since there are only 2048 pixels per column, this means that it requires very little RAM, and the time penalty for projecting out Fourier vectors using Equation 6 is small compared to using the FFT algorithm.[2]

4.1. *Computing Requirements*

When used in the recommended mask mode, NSClean is not computationally demanding. The execution time on our development server is about 6 seconds for one 2048×2048 pixel NIRSpec image. The server, which is a few years old, has 8× Intel Xeon cores running at 3.5 GHz and 250 GB of RAM. In practice, NSClean uses only a tiny fraction of the RAM. Although our server has an NVIDIA Quadro M4000 GPU with 8 GB of RAM, in practice we found that NSClean's execution time was about the same in CPUs as in the GPU. This is because Equation 6 's matrices are not large when images are processed in columns.

[2] The FFT only works for fully sampled data, which we do not have.

We have also tested NSClean on a 2019 MacBook Pro. Execution time on the MacBook is about 12 seconds per image. The MacBook has an 8-Core Intel i9 CPU running at 2.3 GHz and 32 GB of RAM. Again, NSClean did not use much of this RAM. According to the Apple ActivityMonitor App, peak usage was about 150 MB.

The NSClean prototype (the NSClean1 class in the distribution) was computationally intensive. In general, we find that mask mode (the recommended NSClean class) provides better correction and is much less taxing. We have nevertheless left NSClean1 in the distribution in case anybody finds it useful. For NSClean1, using a GPU can provide a roughly a >10× speedup compared to CPUs. Using a GPU, NSClean1's execution time is about 3 seconds. The execution time using the server's CPUs was a minute or two.

Our development server had the following software; Oracle Linux Server release 8.7, python-3.10.8, astropy-5.0.4, cupy-11.5.0, numpy-1.22.3, and pillow-9.3.0.

4.2. *Multithreading*

NSClean is not explicitly multithreaded. In practice, however, we always have multithreading turned on for python's linear algebra libraries. As a result, when we run NSClean, it usually shows all CPUs being used because most of the work is linear algebra.

On our Intel-based computers, this is done by installing the Intel version of numpy and setting an environment variable. For our 8-core server, the python code is as follows.

```python
import os
os.environ['MKL_NUM_THREADS']='8'
```

Our understanding is that on non-Intel computers, similar functionality exists, although the environment variables are different.

When a GPU is used, python's cupy package automatically parallelizes the linear algebra operations over however many GPU cores are available. Our NVIDIA Quadro M4000 has 1664 CUDA cores. Individual CUDA cores are slow compared to the server's CPUs. However, because there are so many of them, they enable a $>10\times$ speedup for NSClean1.

4.3. *Installing NSClean*

NSClean is a standard pip-installable python package. It is available from the NASA JWST website (NASA JWST website 2023). To install it on MacOS or Linux, change into a directory that is in your python path, and download the distribution. Then, use pip to install it,

```
pip install -e nsclean.
```

This will install nsclean as an editable package in your python path.

5. SUMMARY

Many JWST observers are finding that there is faint vertical banding and a picture frame pattern in pipeline calibrated NIRSpec images. The effect is particularly challenging for IFU observations because it can add spectral features that are not real. This article describes the NSClean python package that uses dark areas of NIRSpec scenes to remove this noise. To use NSClean, the astronomer must provide a mask specifying which pixels are to be treated as background. For each count rate image, NSClean then: (1) computes the Fourier transform of the background using an algorithm that can handle missing data, (2) applies a low-pass filter to reduce noise, and (3) inverts the Fourier transform yielding a background model. When the background model is subtracted from the image, it removes most of the correlated noise. NSClean is simple and computationally undemanding. The NSClean python package is freely available for download from the NASA JWST Website (NASA JWST website 2023).

This work was supported by NASA as part of the JWST.

Facilities: JWST(NIRSpec)

Software: astropy (Astropy Collaboration et al. 2013, 2018)

REFERENCES

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068

Astropy Collaboration, Price-Whelan, A., Sip\Hocz, B., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Cooley, J. W., & Tukey, J. W. 1965, Math. Comput., 19, 297, doi: 10.2307/2003354

Jakobsen, P., Ferruit, P., Alves de Oliveira, C., et al. 2022, A&A, 661, A80, doi: 10.1051/0004-6361/202142663

JWST User Documentation website. 2016. https://jwst-docs.stsci.edu/jwst-near-infrared-spectrograph/nirspec-observing-strategies

Loose, M., Farris, M. C., Garnett, J. D., Hall, D. N. B., & Kozlowski, L. J. 2003, Proc SPIE, 4850, 867

NASA JWST website. 2023. https://webb.nasa.gov/content/forScientists/publications.html

Rauscher, B. J. 2015, PASP, 127, 1144, doi: 10.1086/684082

Rauscher, B. J., Arendt, R. G., Fixsen, D. J., et al. 2013, in Proc SPIE, ed. H. A. MacEwen & J. B. Breckinridge, Vol. 8860, International Society for Optics and Photonics (SPIE), 886005, doi: 10.1117/12.2025053

Rauscher, B. J., Arendt, R. G., Fixsen, D. J., et al. 2017, PASP, 129, 105003, doi: 10.1088/1538-3873/aa83fd