Software and methods developed for the FIGS Team

Astrogrism 2020

N. Pirzkal May 2020

A pot-luck of grism related code and methods

- GRISMCONF: A python module and “framework” to deal with grism calibration and dispersion
- SBE: Simulation Based Extraction, an updated look at generating 1D spectra
- WFC3_Back: New background models and new method to perform background subtraction
- EM2D: Extraction-free, naked emission lines search methodology.
- MAP2D: Resolving emission line 2D structure using forward modeling
- JWST implication: Use of SBE to extract Mirage simulations

GRISCONF

My lowest level building block

- All of the code I am presenting here rely in this small module, which was optimized for speed
- GRISMCONF provides access to the transformation between imaging and dispersed frame
    - Relies on calibration files that describe the grism and its field dispersion:
        - Trace (where)
        - Wavelength (what)
- Grism configurations for ACS and WFC3 were in ‘aXe’ format. These contain 2D polynomial descriptions of the traces and wavelength dispersion
- aXeConf was created to provide access to these to Python code.
- Encoding/calibration scheme was revised in 2017, introducing GRISMCONF. Essentially similar to aXeConf but allowing for the calibration to be encoded in a more flexible manner and provides easier reverse operations ((x,y,λ) to (xg,yg) as well as (xg,yg) to (x,y,λ) .
- See ISR WFC3 2017-01 for full details.
- Personal implementation is available at https://github.com/npirzkal/GRISMCONF
- GRISMCONF configuration files for WFC3 G102/G141, NIRCAM, and NIRISS currently exist.
- This is one of the most used routine for my code, so it is designed to be fast

$$\hat{x} = x' - x = f_x(x, y; t) \tag{3a}$$
$$\hat{y} = y' - y = f_y(x, y; t) \tag{3b}$$
$$\lambda = f_\lambda(x, y; t) \tag{3c}$$

$$t = f_x^{-1}(x, y; \hat{x}) \tag{4a}$$
$$t = f_y^{-1}(x, y; \hat{y}) \tag{4b}$$
$$t = f_\lambda^{-1}(x, y; \lambda) \tag{4c}$$

In the older aXe representation, eq. 3c and 4c used the path length along the trace instead of the more generalized variable t, which introduced an inversion problem since the pathlength is:

$$\Delta s(x, y, \Delta x) = f(\Delta x) = \int_0^{\Delta x} \sqrt{1 + (\frac{dy}{dx})^2} dx$$

GRISMCONF

Basic examples

- Extracting:
  - We want to compute the wavelength of a pixel along the trace of a dispersed image:

```
import grismconf

C = grismconf.Config("G102.conf")

dx = xp - x0
t = C.INVDISPX("A",x0,y0,dx)
wavelength = C.DISPL("A",x0,y0,t)
```

- Simulating/Dispersing:
  - We want to compute the pixel at which light at a given wavelength falls on the dispersed image:

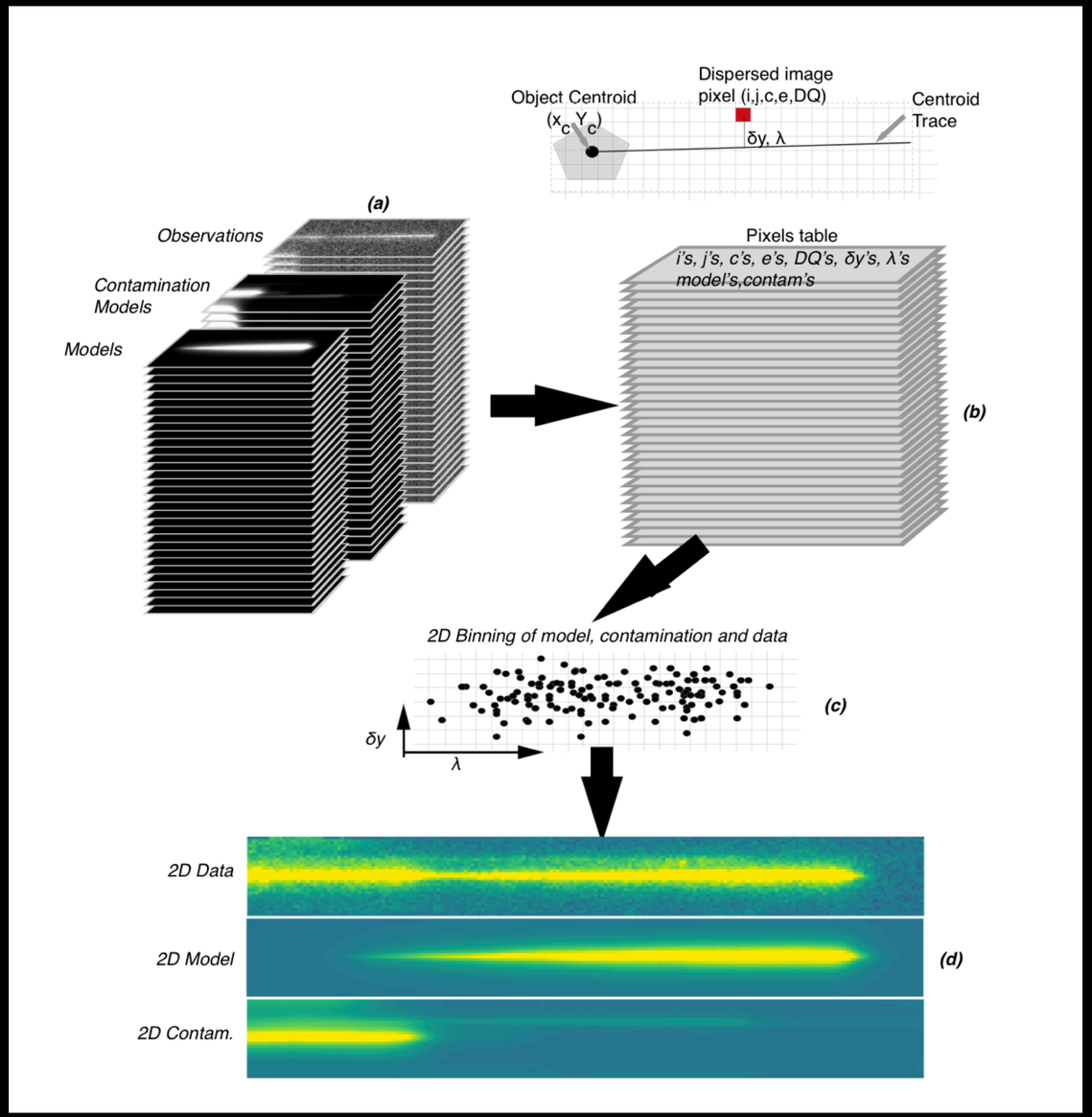
```
import grismconf

C = grismconf.Config("G102.conf")
t = C.DISPL("A",x0,y0,wavelength)
dx = C.DISPX("A",x0,y0,t)
dy = C.DISPY("A",x0,y0,t)

xp = x0 + dx
yp = y0 + dy
```

Goal: Providing as much flexibility as possible to the user, as well as the person calibrating the instrument, to go from direct image reference frame to dispersed image reference frame, and vice versa.

SBE

Simulation Based Extraction

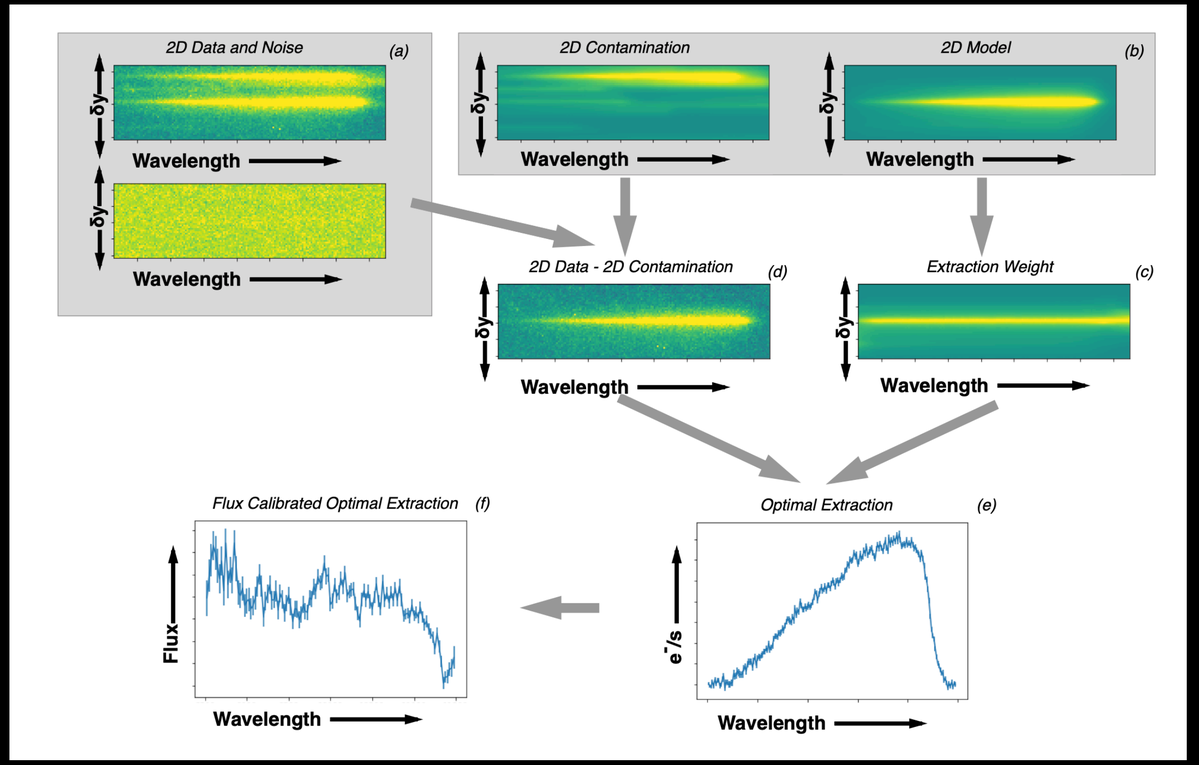
- Why a new name?
  - Unlike aXe, SBE is designed to be driven by our understanding of the data, i.e. simulations and not simply catalogs. aXe has evolved over the years to implement most of the SBE approach but it is a bit added-on.
  - The idea is that if we can come close to simulating the data, even before extracting it, then we are in good shape to do a careful extraction
  - Simulations are crucial:
    - To provide a mask for each dataset to mask out spectra to estimate the background levels of each dataset
    - To estimate the contamination level in each extracted (2D or 1D) spectrum
    - To provide (cross-dispersion) extraction weights for each individual spectrum to perform optimal extraction
    - To provide a customized sensitivity function for each object that accounts for the size/blurring effect caused by the extended size of sources
  - Advantage of SBE is that it provides a step by step approach that allows one to continuously check each intermediate step of the process (i.e. avoid the big black box problem)
  - SBE requires short direct images to be taken right before or after a grism observation



SBE

Simulation Based Extraction

- Basic steps:
  - Step 1: Aggregate all the knowledge we have about the field and individual sources (catalogs, morphologies, spectral energy distribution from imaging, a-priori spectra, etc…)
  - Step 2: Fix any problems with the astrometry:
    - using external catalog/mosaic if possible (i.e. attempt to fix the astrometry in an absolute sense)
    - using self consistency if no external info is available (i.e. make a mosaic using direct imaging, fix relative offsets between observations)
  - Step 2: Simulate individual observations (FLT files in case of HST, rate files in the case of JWST)
  - Step 3: Model and subtract the dispersed background (more on this later)
  - Step 4: Basic extraction, which is essentially just book keeping of all the information about dispersed images pixels, count rates, and wavelengths.
  - Step 5: (this is where things diverge from forward modeling methods, more on this later) Assemble 2D, `rectified' spectra, which can be thought of as having wavelength on the x-axis and cross-dispersion distance on the y-axis, which are background subtracted and contamination corrected. Multiple observations are combine at this time.
  - Step 6: Produce 1D extracted, `classic' spectra from 2D stamps, potentially using an optimal extraction scheme (XXX) using the data themselves (in case of high S/N) or our simulations.



- See Pirzkal+ 2017 (https://ui.adsabs.harvard.edu/abs/2017ApJ...846...84P/abstract) for a description of the full process when applied to deep G102 observations

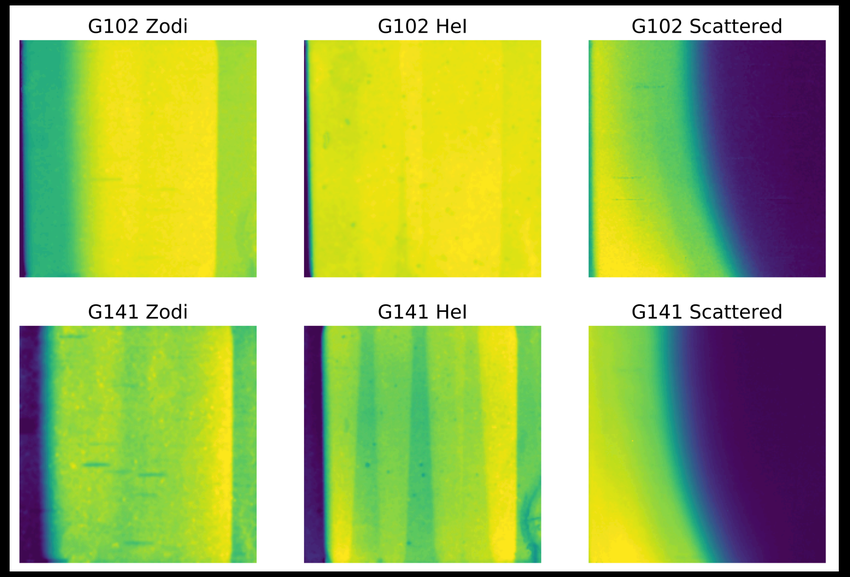
WFC3_Back

Grism background subtraction

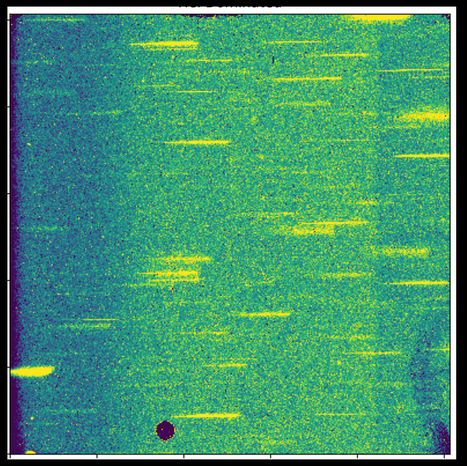
- Background can be subtracted locally, estimating the background above and below a virtual `aperture' BUT:
  - This does not work for even moderately crowded fields
  - Can introduce artifacts (interpolation dependent)
  - In the case of WFC3, fails to deal with varying background, which is crucial as on-the-ramp and CR rejection is then not possible

- In the case of WFC3, which suffers from a high level of background compared to ACS, we have estimated full frame dispersed imaged of the Zodi (constant), Hel (varying) and Scatter (varying) components.
- The components and code is described in WFC3 ISR 2020-04, and it released officially as part of the WFC3tools python package
- My own version is also available on Github at https://github.com/npirzkal/WFC3_Back_sub

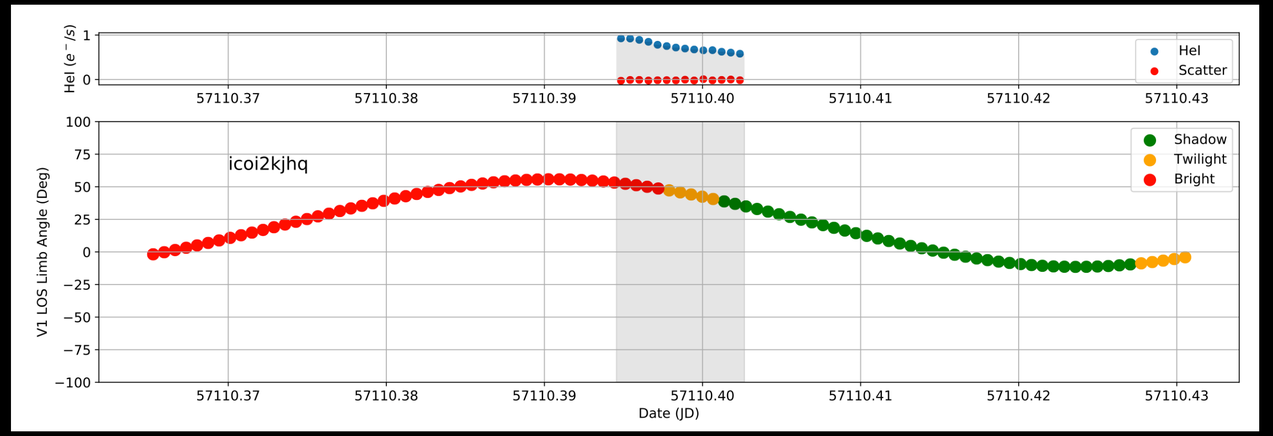
Models



Observation



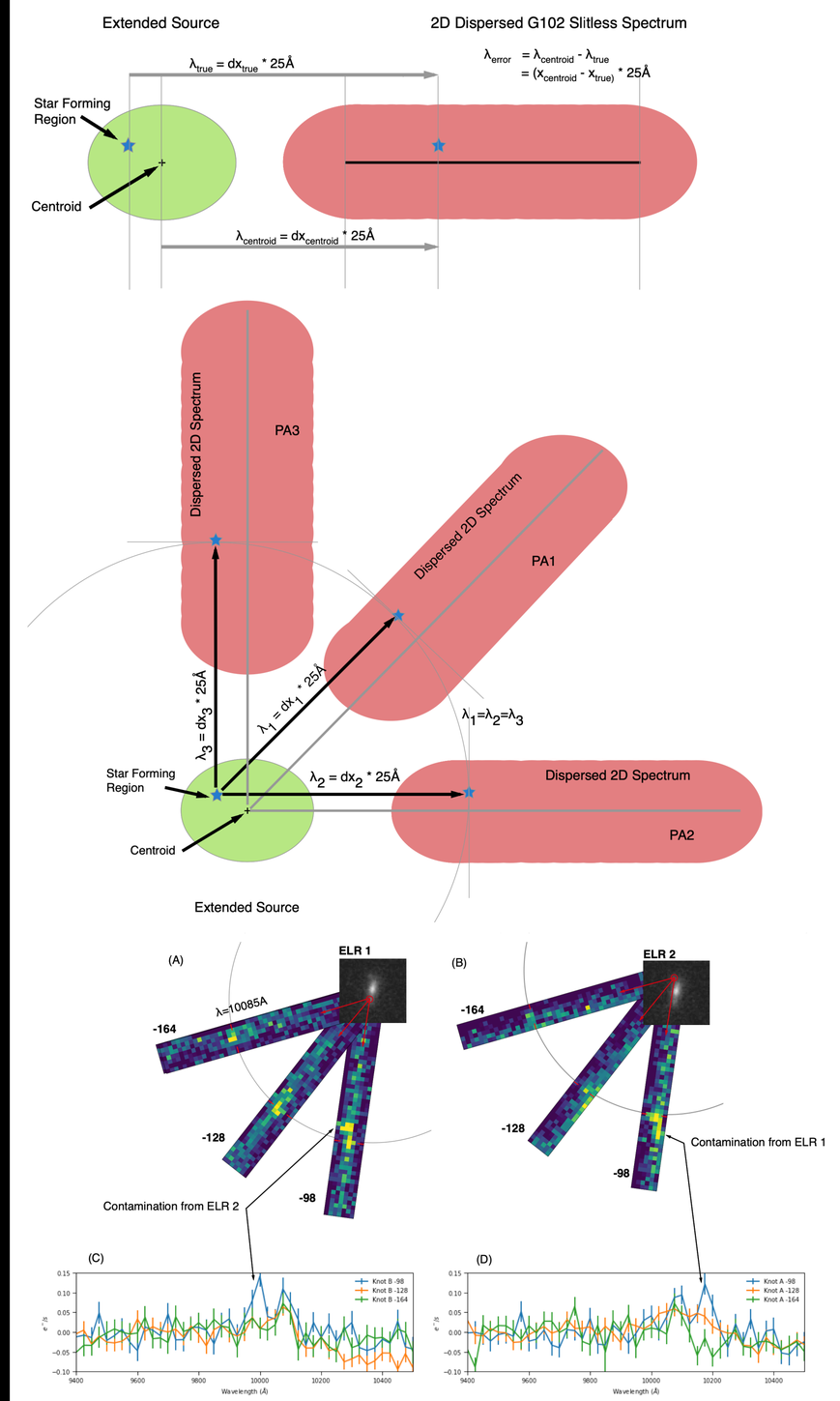
$$\mathcal{S}_{p,i} = z\mathcal{Z}_p + b_i\mathcal{B}_p + e_i\mathcal{E}_p$$



EM2D

Not everything is best done by extracting and fitting spectra

- A lot of the available packages provide a `turn key' way to extract spectra and/or estimate redshifts
- There are cases where different approaches are desired, usually science driven.
- One example: EM2D
  - We want look for emission lines directly, without extracting data at all
  - Emission line regions DO not match the broad band footprints of objects
  - Relies on two things:
    - Good calibration of the instrument (took several years for WFC3)
    - GRISMCONF
  - Improves wavelength calibration of the line (since they can be offset from the centroid of the sources)
  - EM2D identifies the source of emission line in the imaging plane to the limit of the calibration as well as determine accurate wavelengths for these emission lines
- Method is fully described in Pirzkal+ 2018 and was applied to the deep FIGS G102 observations.



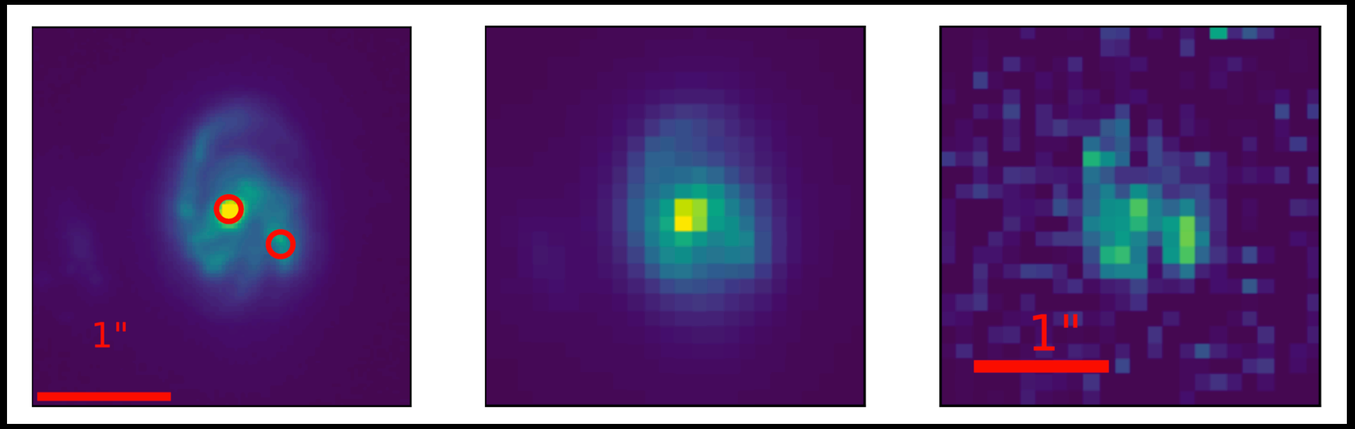
MAP2D

Not everything is best done by extracting

- There are instances when forward modeling is also warranted
- Example:
  - We used EM2D to find emission line knots, we know the exact wavelengths of these line
  - We want to create 2D maps in those narrow wavelengths
- We can **forward model** a model to match our individual observations (since we have access to something like GRISMCONF)
  - We can either model continuum subtracted observations at a finite number of wavelengths, or attempt to construct a full 3D cube.
- Forward modeling approach has the advantage of dealing with noise and nuisance parameters (for example left over background levels, wavelength errors) properly (unlike a reverse approach such as drizzling things back).
- Examples are shown in Pirzkal+ 2018

- V. 1.0 initially approached the problem as a likelihood minimization problem using GRISMCONF and MPfit
- V. 2.0 re-factored the problem as a linear algebra problem, similarly to the LINEAR (Ryan+ 2018) approach and is significantly faster. The W matrix is built using GRISMCONF
  - In this approach we treat each pixel separately and solve for $f_{\lambda,j}$

$$G_{x,y,i} = \sum_{\lambda} \sum_{j}^{N_{pix}} W_{x,y,i,\lambda,j} f_{\lambda,j}$$



JWST SBE Extraction

NIRCAM example

- Relying on a module such as GRISMCONF, extracting data using SBE is relatively easy
- We have created Jupyter notebooks to do this for NIRCAM or NIRISS
- A simulation of a deep galaxy field was created using Mirage
- Mirage was also used to generate the noise free simulation of each source
- SBE was implemented in a 100+ lines of code
- Available at https://github.com/npirzkal/MirageExtract