20250511 ppxf mass to light

1. Data

I am still using IC3392

```
Cube dimensions  →  nz = 3761,  ny = 438,  nx = 437
```

2. Wavelength cutoff and velocity scale

''Sky subtraction is clearly not perfect, but the best that we can do for the moment. Below $7000Å$ it is generally acceptable, at longer wavelengths the situation is worse.''

So I make a cutoff at $\sim 7000Å$. Actually, now it remains $4750 - 7050Å$:

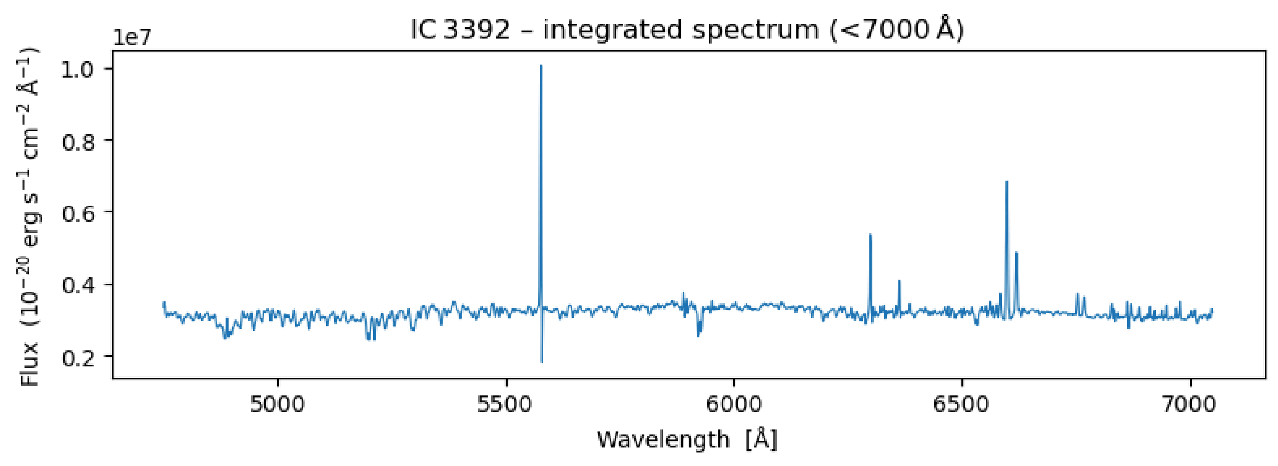
```
lam_min = 4750.0                          # min λ in Å
lam_max = 7050.0                          # max λ in Å
```

Then I compute the velocity scale:

```
c_kms    = c.c.to(u.km/u.s).value         # 299 792.458
dlnλ     = np.diff(np.log(lam_ang))   # dlnλ in Å
velscale = np.min(c_kms * dlnλ) # km/s per pixel
```

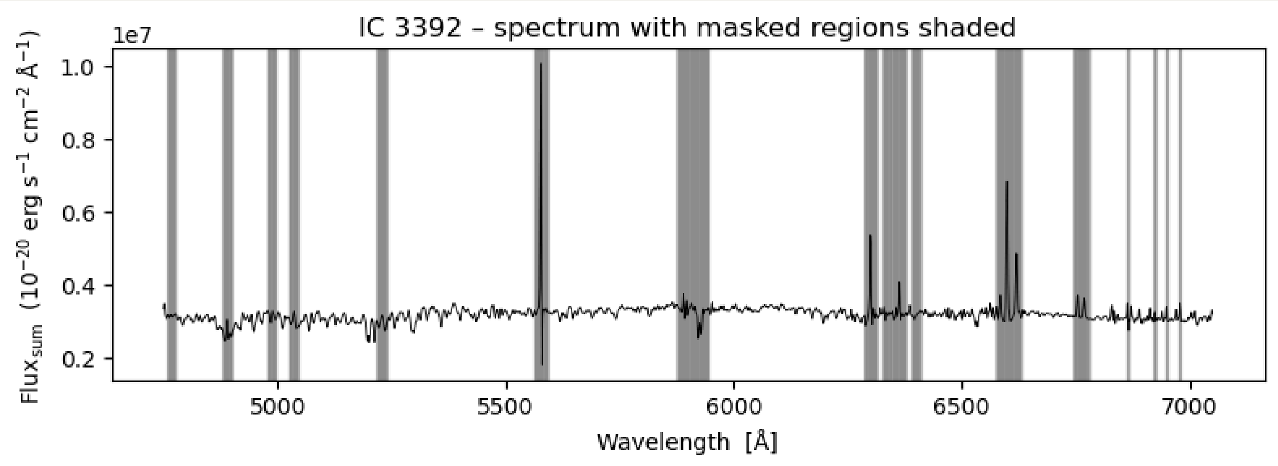
This gives $53.16km/s$.

Here is the native spectrum:



3. Mask emission lines

Since we are only interested in the continuum, I mask the emission lines from galaxy (observer frame) and air (rest frame) by `specMask_KIN.txt`. I just mask them without modifying the raw spectrum:



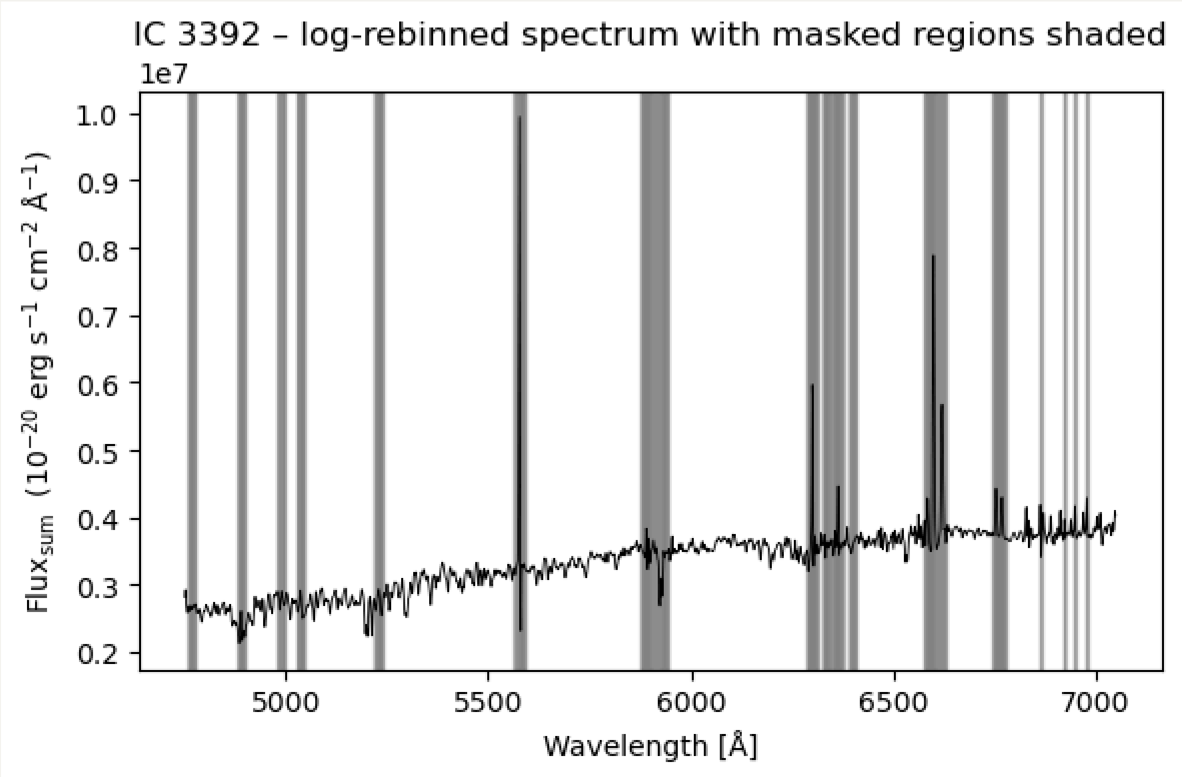
Note: "Because the MUSE spectrographs do not operate in vacuum the wavelength calibration is based on arc line wavelengths in standard air (Weilbacher et al. 2020)." Thus, it is correct that we are taking the lines measure in air.

4. `log_rebin`

Now I need to do `log_rebin`. It seems that this is one of the requirements in `pPXF`. But the question is that, why in natual log rather than $\log_{10}$ (same question for `velscale`)? No need to multiply an extra constant when taking derivative?

I still do `log_rebin` anyway for both flux and noise, and I force `velscale=velscale`, so I got:

```
Log-grid length : 2228 pixels
velscale        : 53.159 km/s
```



5. SPS templates: E-MILES

Then I load the SPS templates. Here I choose `spectra_emiles_9.0.npz` because it seems to be more suitable for IFS data.

E-MILES SPS model templates: Vazdekis et al. (2016).

6. FWHM and MUSE LSF

`pPXF` requires thestellar templates and the galaxy spectrum to have the same instrumental resolution before it adds any extra broadening for the LOSVD.

Emsellem+2022 use this equation for MUSE LSF:

$$FWHM\left(\lambda\left[\AA\right]\right) = 5.866 \times 10^{-8}\lambda^2 - 9.187 \times 10^{-4}\lambda + 6.040.$$

The idea is that the templates should have the same FWHM as muse, so:

```
lam_temp_arr = np.arange(lam_min, lam_max+1, dtype=int)
```

```
fwhm_gal = {
    "lam": lam_temp_arr,
    "fwhm": 5.866e-8 * lam_temp_arr**2
            - 9.187e-4 * lam_temp_arr
            + 6.040
}

sps = lib.sps_lib(
    filename, velscale_out, fwhm_gal,
    norm_range=[lam_min, lam_max],
    lam_range=[3e3, 1e4],
)
```

7. `pPXF` fitting

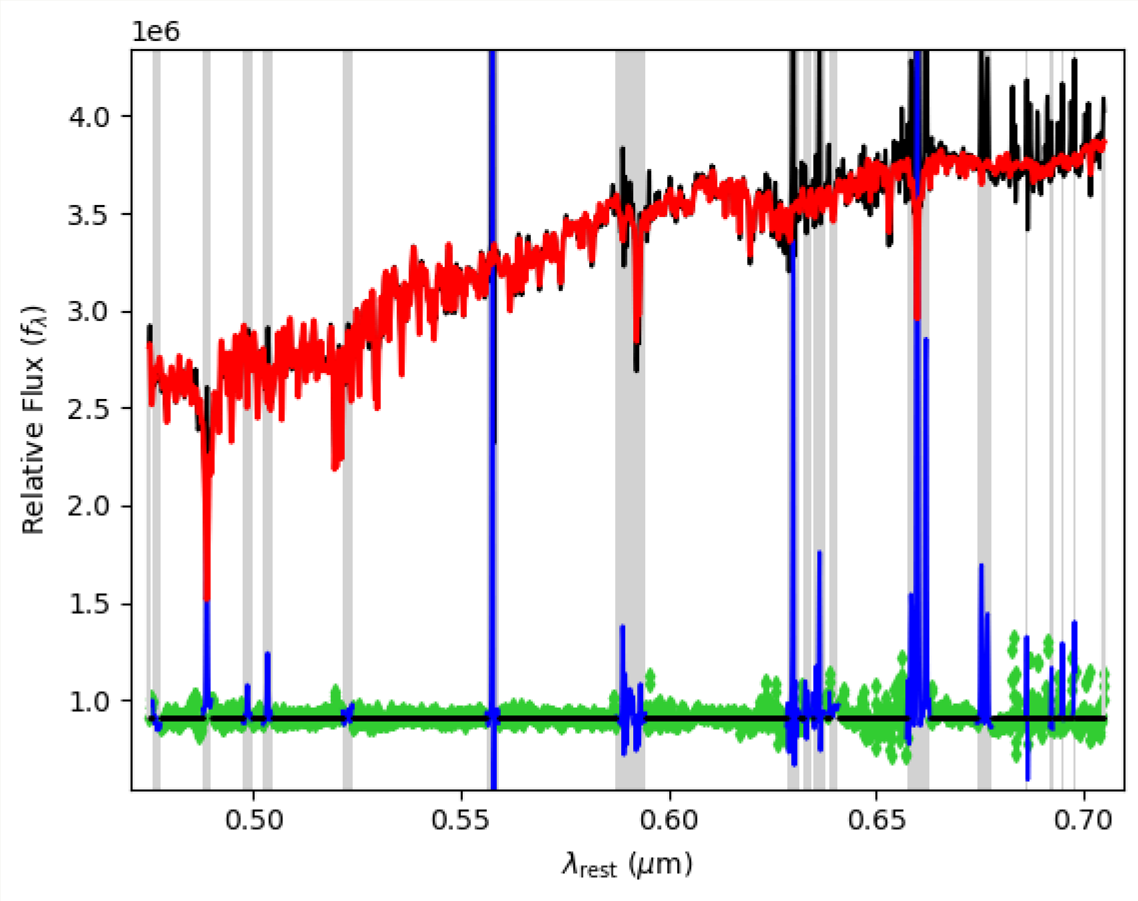
By passing the mask regions as the `goodpixels`, we can run `pPXF` (I choose to include regulization):

```
reg_dim = sps.templates.shape[1:]
stars_templates = sps.templates.reshape(sps.templates.shape[0], -1)
regul_err = 0.1  # Large regularization error

# Compute which rebinned pixels fall on unmasked (good) λ
lam_grid = np.exp(log_lam)                   # rebinned wavelengths
in Å
pix      = np.searchsorted(lam_ang, lam_grid)
goodpixels = np.where(~mask_bad[pix])[0]

# Run pPXF only on those good pixels
from ppxf import ppxf

start_V    = v_guess
start_sig = 5 * velscale_out

pp = ppxf.ppxf(
    # templates    = sps.templates,
    templates    = stars_templates,
    galaxy       = log_flux,
```

```
    noise         = log_noise,
    velscale      = velscale_out,
    start         = [start_V, start_sig],
    degree        = 12,
    mdegree       = 0,
    moments       = 2,
    # trig          = 1,
    # clean         = True,
    goodpixels    = goodpixels,
    lam           = np.e**(log_lam),
    lam_temp      = sps.lam_temp,
    regul         = 1/regul_err,
    reg_dim       = reg_dim,
    plot          = True)

print(f"V = {pp.sol[0]:.3f} km/s,  σ = {pp.sol[1]:.3f} km/s")
```

Note that "Unless a good initial guess is available, it is recommended to set the starting sigma >= 3*velscale in km/s (i.e. 3 pixels)." Here I choose initial guess to be `start_V` as my estimation when matching the emission line at observer frame, and `start_sig` to be 5 times of velcity scale. Fitting model is set as an additive polynomial of 12 with no multiplicative polynomia and first 2 moments of Gauss–Hermite expansion. This yields:

```
 Best Fit:       Vel     sigma
 comp.  0:      1675        56
chi2/DOF: 913.9; DOF: 1833; degree = 12; mdegree = 0
method = capfit; Jac calls: 4; Func calls: 14; Status: 2
linear_method = lsq_box; Nonzero Templates (>0.1%): 118/150
V = 1675.283 km/s,  σ = 56.196 km/s
```



8. Mass-to-Light ratio

Now with the fitting results, I can extract the weight to get M/L. Since we pick wavelength within $4750 \sim 7000 \AA$, I choose the M/L in r band of SDSS.

```
z_fit = pp.sol[0] / c_kms
print(f"Fitted redshift  : {z_fit:.5f}")
# # ---------- Composite M/L from pPXF weights --------------------
----
weights = pp.weights.reshape(reg_dim) / pp.weights.sum()
ML_r = sps.mass_to_light(weights, band="SDSS/r", redshift=z_fit,)
                    # intrinsic M/L (dimensionless)
print(f"M/L (r band)  : {ML_r:.3f}  M⊙/L⊙ =
log({np.log10(ML_r):.3f}  M⊙/L⊙)")
```

This returns:

```
Fitted redshift  : 0.00559
(M*/L)=2.677 (SDSS/r at z=0.0056)
M/L (r band)  : 2.677  M⊙/L⊙ = log(0.428  M⊙/L⊙)
```

Now I can also find the r band luminosity and compute the total stellar mass:

```
D = 11.5 * u.Mpc           # Virgo distance

# 0) r-band data
lam  = lam_ang * u.angstrom
F_λ  = flux_sum * 1e-20 * u.erg/u.s/u.cm**2/u.angstrom
mask_r = (lam_ang> 5400) & (lam_ang < 7000)
lam_r = lam[mask_r]
F_λ_r = F_λ[mask_r]

# 1) Convert to Fv
F_v = (F_λ_r * lam_r**2 / c.c).to(u.erg/u.s/u.cm**2/u.Hz)

# 2) Sort and integrate over v
v = (c.c / lam_r).to(u.Hz)
idx = np.argsort(v)
v_sorted, Fv_sorted = v[idx], F_v[idx]

Fv_int = np.trapezoid(Fv_sorted, v_sorted)
Δv     = v_sorted.max() - v_sorted.min()
Fv_avg = Fv_int / Δv

# 3) AB zero-point
Fv0 = (3631 * u.Jy).to(u.erg/u.s/u.cm**2/u.Hz)

# 4) AB magnitude
m_r = -2.5 * np.log10(Fv_avg / Fv0)
print(f"AB apparent magnitude = {m_r:.3f}")

# 5) Compute absolute magnitude
M_r = m_r - 5 * np.log10(D.to(u.pc).value / 10)

# 6) Solar AB magnitude in r-band
M_sun_r = ppxf_util.mag_sun(bands="SDSS/r", redshift=z_fit,
system="AB")[0]

# 7) Luminosity in solar units
L_r_sun = 10**(-0.4 * (M_r - M_sun_r))
print(f"Absolute M_r = {M_r:.2f}, M_sun,r = {M_sun_r:.2f}")
```

```
print(f"r-band luminosity = {L_r_sun:.3e} L_sun =
log({np.log10(L_r_sun):.3f} L⊙)")

# 8) Get the physical luminosity in erg/s:
L_r = L_r_sun * c.L_sun
print(f"r-band luminosity = {L_r:.3e}")

# 9) Given that we know the mass-to-light ratio, we can compute the
stellar mass
M_star = ML_r * L_r_sun
print(f"Log M_star          : log({np.log10(M_star):.3f}  M⊙)")
```

Finally, we have

```
AB apparent magnitude = 12.369
Absolute M_r = -17.93, M_sun,r = 4.65
r-band luminosity = 1.077e+09 L_sun = log(9.032 L⊙)
r-band luminosity = 4.121e+35 W
Log M_star          : log(9.460  M⊙)
```

Note. At first, I forgot to normalize the SPS templates by using `norm_range`, and that leads to $M_*/L_r = 8$ last Thursday. Later on I add `norm_range` but this still gives unreasonable value ~ 0.9. Just today I realized that I shouldn't further `sps.templates /= np.median(sps.templates)`, otherwise this will affect the weighting in `pPXF` and thus yields wrong M_*/L_* ratio. Also by correctly set up normalization of SPS templates, it can resolve the fitting issue at H_β absorption.

I also show the weights fraction here: