Quantum-yield high and low with large spot size →

High and low powers measured 2 spot sizes using lenses and flip mirror plus beam splitter to attenuate the light.

Author: Jean Matias
UCNP Sample: s013
spot sizes: 500um, 100um
Dye Sample: s017
Acquired on: 2021/04/30
Analysed on: 2021/05/04 (May the fourth be with you)
Last Updated: 2022/03/01

- Metadata available at the end of the notebook,
- The final data that generates the plots and their metadata are saved at the end of the notebook in ../data/analysed-data/ folder.

In [1]:

```
%matplotlib notebook
import sys
sys.path.append('../scripts/')
from lab import Sample, Analysis, BeamProfile
from calibration import *
```

```
For details on the analysis protocol and calculation:
>>> help(Analysis)
>>> help(Sample)
```

In [21]:

```
help(Analysis)
```

```
 |      1. Get steady data - it averages data points after the equipmment
and samples stabilised.
 |      2. Calibrate the power meter output (pm) - Volts to mW using stand
ard curves for that.
 |          This is the transmitted power through the cuvette.
 |      3. Calibrate the DAQ output (trigger) - Volts to mW using empty me
asuremnts.
 |          This is the laser power, i.e. the power before reaching the cu
vette.
 |      4. Get the power density in the center - It uses the power before
and after the cuvette
 |          along the beam width.
 |      5. Get absorbance - Sample-Absorbance is the abosorbance of the wh
ole sample.
 |          On the other hand Absorbance is the absorbance of the compound
without its diluter.
 |          It's necessary to have the diluter abosrobance in order to cal
culate this property.
 |      6. Get absorbed power - Using absorbance it calculates the total p
ower absorbed by the
```

In [3]:

```
help(Sample)
```

```
 |  Methods defined here:
 |
 |  __init__(self, path, expId, sampleType)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  absorbance(self, diluterAbsorbance=None, filterQuery='laser_power >
0', recalculate=False)
 |      Calculates absorbance of the emitter compound present on the aqueo
us medium.
 |
 |      Args:
 |          diluterAbsorbance (float): Diluter absorbance used as a refere
nce to calculate the emitter's absorbance
 |          recalculate (bool): If True it will recalculate the column
 |
 |      Returns:
 |          pandas.DataFrame, float, float, str: Data, Absorbance mean, st
d and success, failure message.
 |
 |  absorbedpower(self, region=[0.45, 0.55], useMean=True, filterQuery='la
```

Selecting samples:

1. ucnps
2. diluters: e.g. water
3. empties
4. dyes
5. diluters: e.g. ethanol
6. empties

In [2]:

```
on20210504 = Analysis.datalog['exp_id'].str.contains('20210331') & ~Analysis.datalog['saved
    & Analysis.datalog['extra_params'].str.contains('sample=(?:s013|empty|water)/.*976nm')
Analysis.datalog[on20210504]
```

Out[2]:

	exp_id	saved_name	out_ch	range_start	range_end	range_step_size	step_reset
645	20210331-090934	./data/raw-data/qy_20210331_090934.csv	ao0=Laser	0.0	5.4	0.08	False
646	20210331-091102	./data/raw-data/qy_20210331_091102.csv	ao0=Laser	0.0	5.4	0.08	False
647	20210331-091239	./data/raw-data/qy_20210331_091239.csv	ao0=Laser	0.0	5.4	0.08	False
648	20210331-091659	./data/raw-data/qy_20210331_091659.csv	ao0=Laser	0.0	5.4	0.08	False
650	20210331-	./data/raw-	ao0=Laser	0.0	5.4	0.08	False

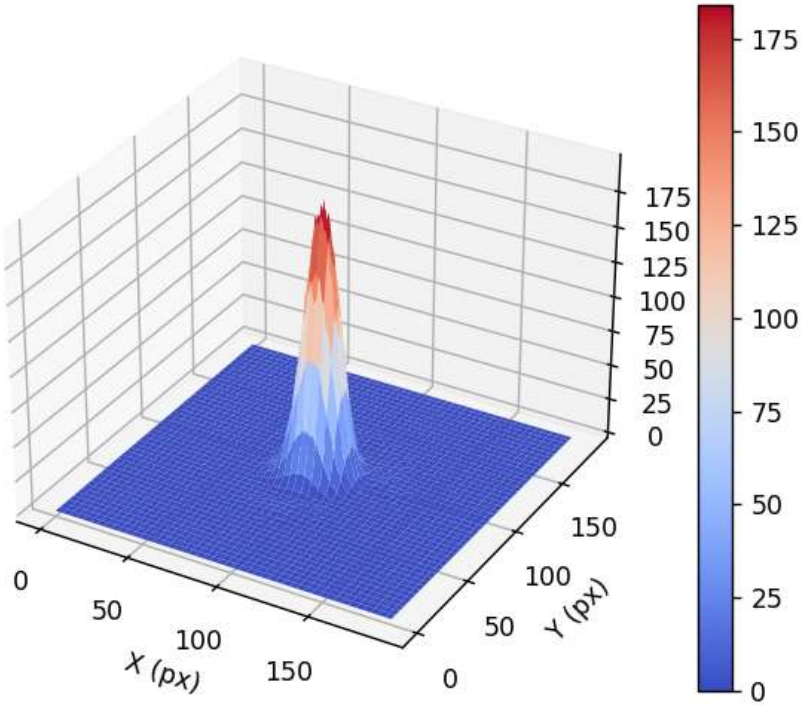
Beam Profile analysis

In [3]:

```
bp2 = BeamProfile('s2_20210430_976nm_140mA_ND1_ET03.png')
bp2_01 = BeamProfile('s2_20210430_976nm_180mA_ND1_BS50_ET03.png')

bp1 = BeamProfile('s1_20210430_976nm_100mA_ND1_ET002.png')
bp1_01 = BeamProfile('s1_20210430_976nm_100mA_ND1_BS50_ET003.png')

bp1.trim(4.5)
bp1_01.trim(4.5)
bp2_01.trim(1.5)
bp2.trim(1.2)

bp1.removebackground(2)
bp1_01.removebackground(2)
bp2_01.removebackground(3)
bp2.removebackground(3)
```

In [4]:

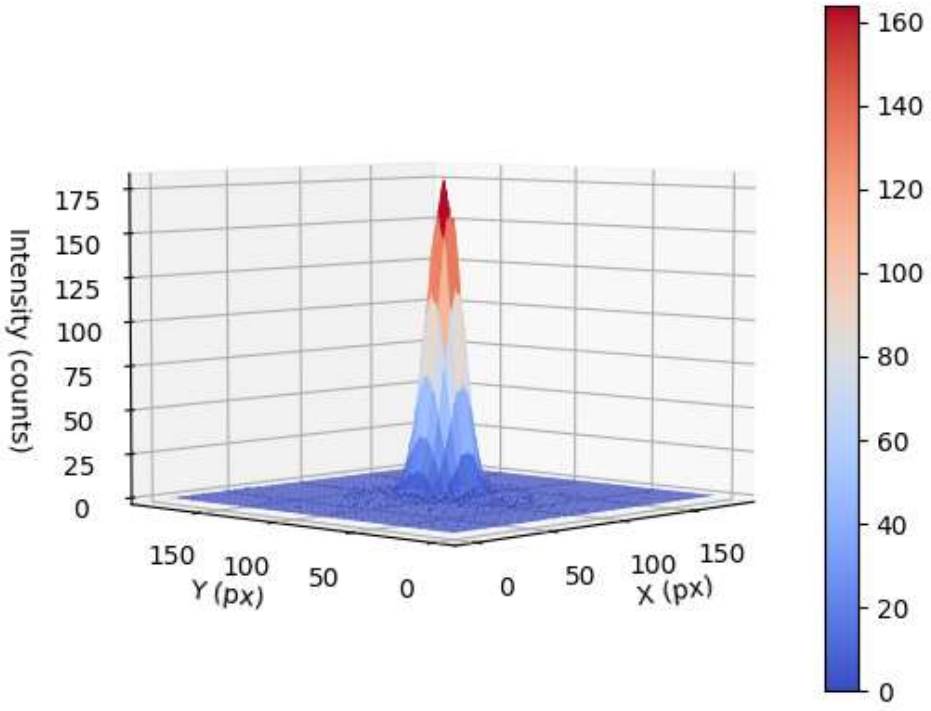
```
bp1.plotsurf()
```



In [11]:

```
bp1_01.plotsurf()
```

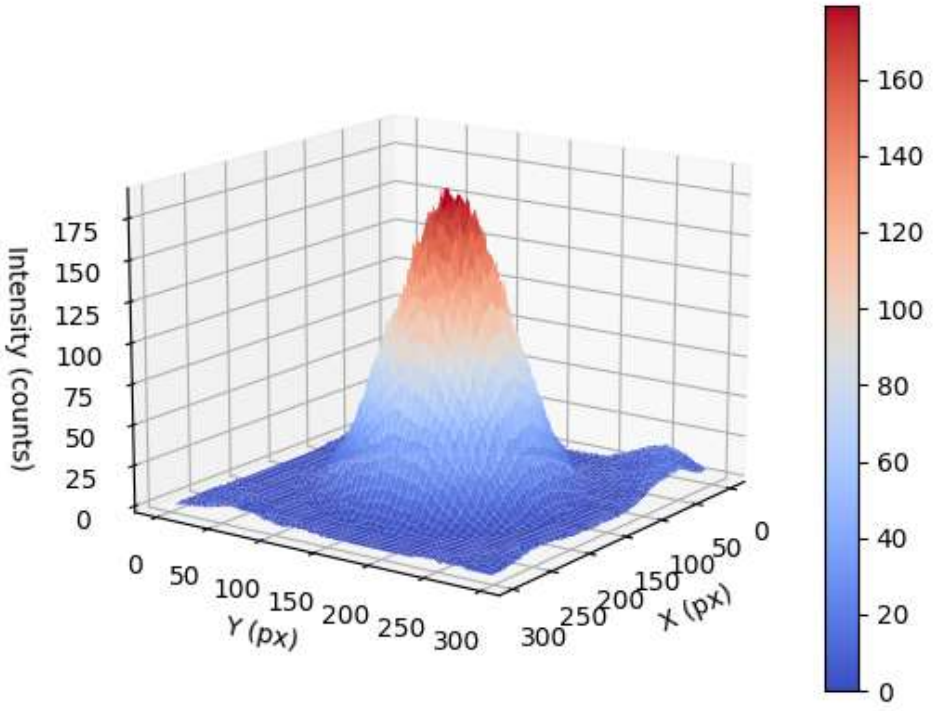
```
<IPython.core.display.Javascript object>
```



In [6]:

```
bp2_01.plotsurf()
```

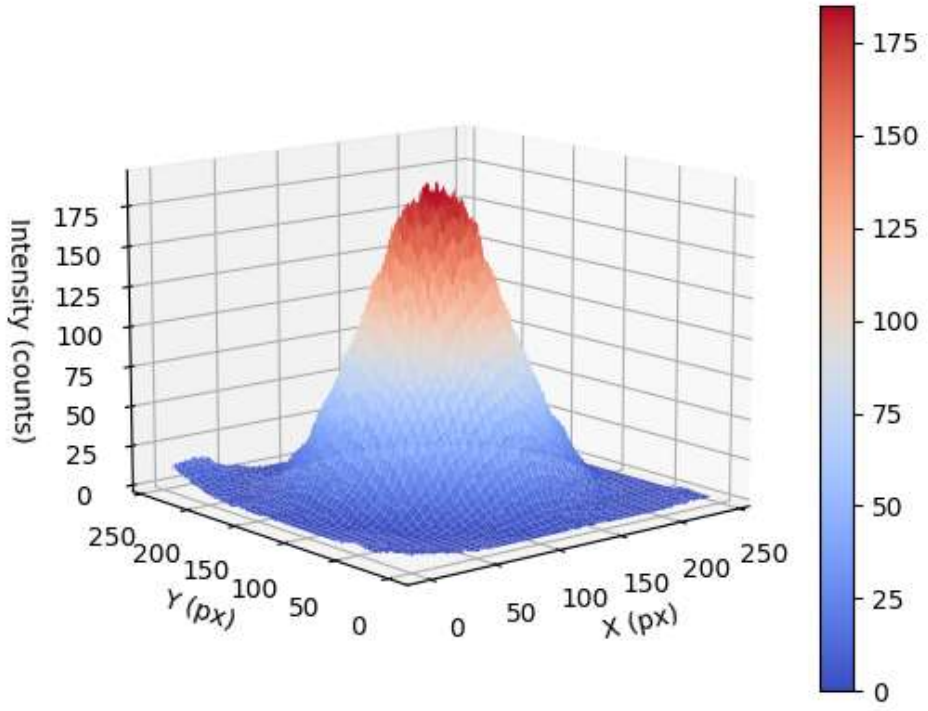
```
<IPython.core.display.Javascript object>
```



In [8]:

```
bp2.plotsurf()
```

```
<IPython.core.display.Javascript object>
```



In [4]:

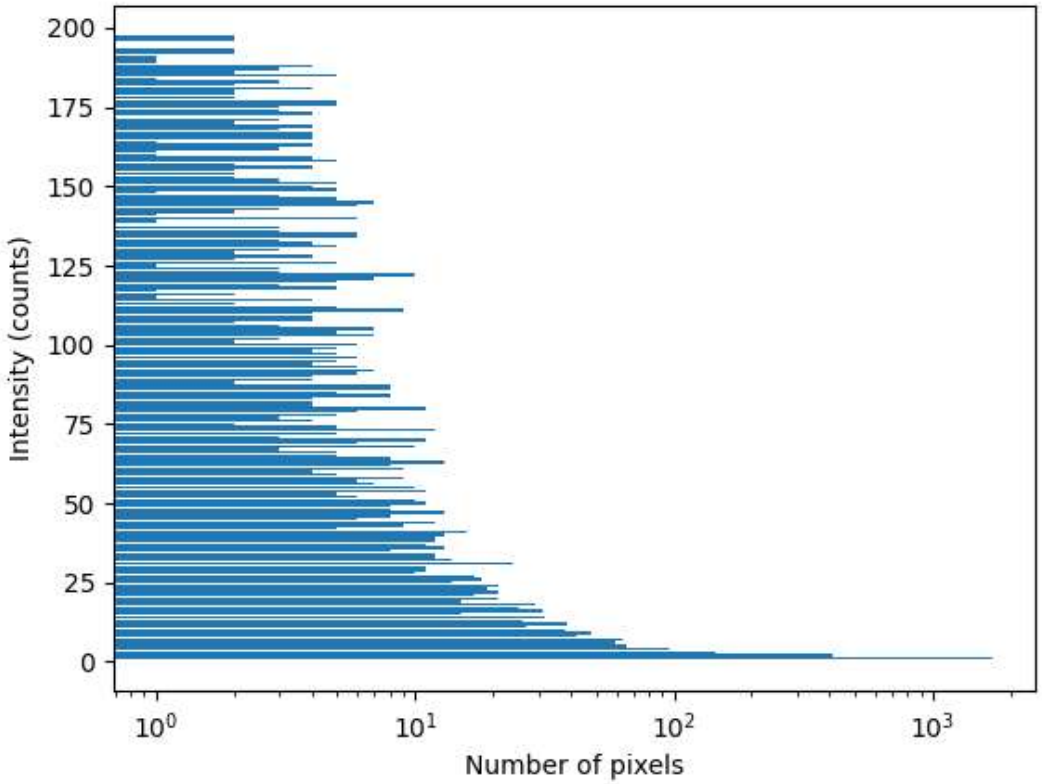
```
print(f'BP1: {bp1.beamwidth():.5f}, BP1 01: {bp1_01.beamwidth():.5f}, BP2: {bp2.beamwidth()
```

```
BP1: 0.01113, BP1 01: 0.01060, BP2: 0.05300, BP2 01:0.05221 in cm
```

In [15]:

```
_ = bp1.histprofile(show=True)
```

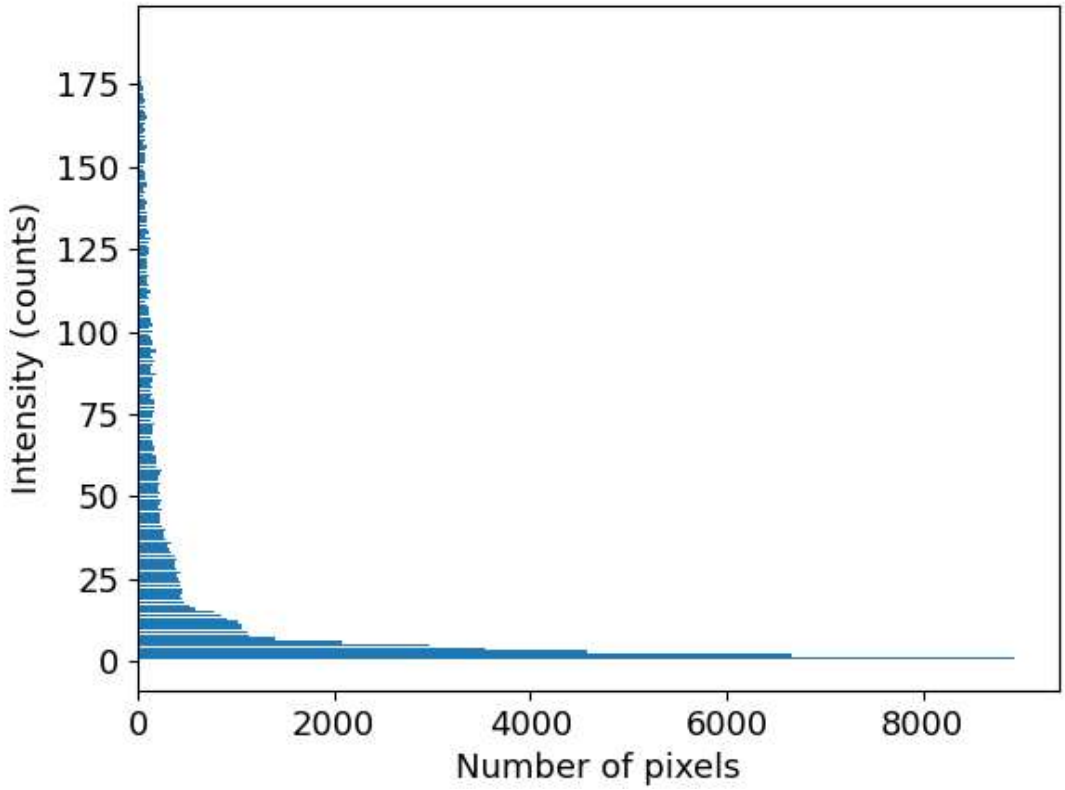
```
<IPython.core.display.Javascript object>
```



In [73]:

```
_ = bp2_01.histprofile(show=True)
```

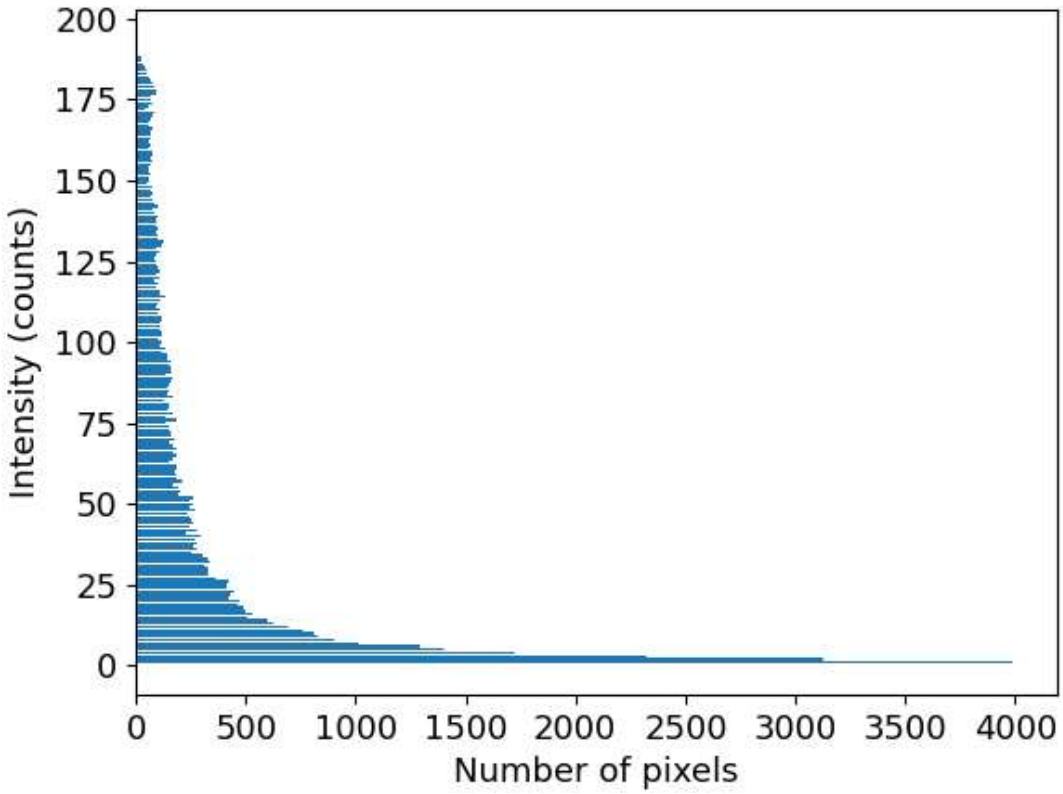
```
<IPython.core.display.Javascript object>
```



In [74]:

```
_ = bp2.histprofile(show=True)
```

```
<IPython.core.display.Javascript object>
```



UCNP data

In [5]:

```
uData = Analysis()
```

In [6]:

```
# Beam S. size = s1 ~100um
uData.loaddata('ucnp', dataIds='20210430-120627', diluterId='20210430-120923', emptyId='202
# Beam S. size = s1 ~100um BS50:50
uData.loaddata('ucnp', dataIds='20210430-120134', diluterId='20210430-120020', emptyId='202

# Beam S. size = 400um
uData.loaddata('ucnp', dataIds='20210430-113345', diluterId='20210430-113227', emptyId='202
# Beam S. size = 400um BS50:50
uData.loaddata('ucnp', dataIds='20210430-114054', diluterId='20210430-114329', emptyId='202
```

In [7]:

```
uData.get('setbeamprofile', dict(beamprofileObj=bp1_01), which=[0, 1, 2])
uData.get('setbeamprofile', dict(beamprofileObj=bp1_01), which=[3,4,5])

uData.get('setbeamprofile', dict(beamprofileObj=bp2), which=[6, 7, 8])
uData.get('setbeamprofile', dict(beamprofileObj=bp2_01), which=[9,10,11])
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

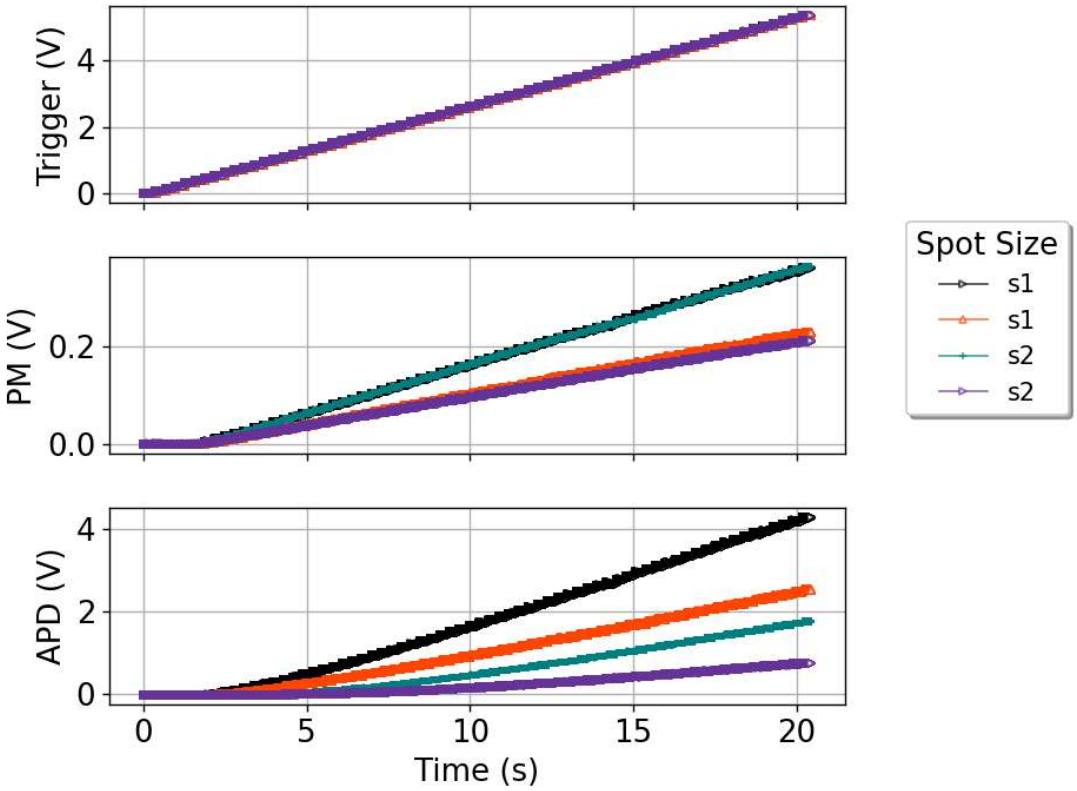
In [10]:

```
uData.details()
```

3	115026	empty	empty	min	instead ND	BS50	976nm	600mW
4	20210430-120020	diluter	water	min	BS50:45 instead ND	BS50	976nm	600mW
5	20210430-120134	ucnp	s013	min	BS50:45 instead ND	BS50	976nm	600mW
6	20210430-112928	empty	empty	min	realigned	0	976nm	600mW
7	20210430-113227	diluter	water	min	realigned	0	976nm	600mW
8	20210430-113345	ucnp	s013	min	realigned	0	976nm	600mW
9	20210430-114220	empty	empty	min	BS50:45 instead ND	BS50	976nm	600mW
10	20210430-114329	diluter	water	min	BS50:45 instead ND	BS50	976nm	600mW
11	20210430-				BS50:45			

In [8]:

```
fig, axs = uData.view(includeDiluter=False, includeEmpty=False, label='spot_size')
```



Get Steady Data

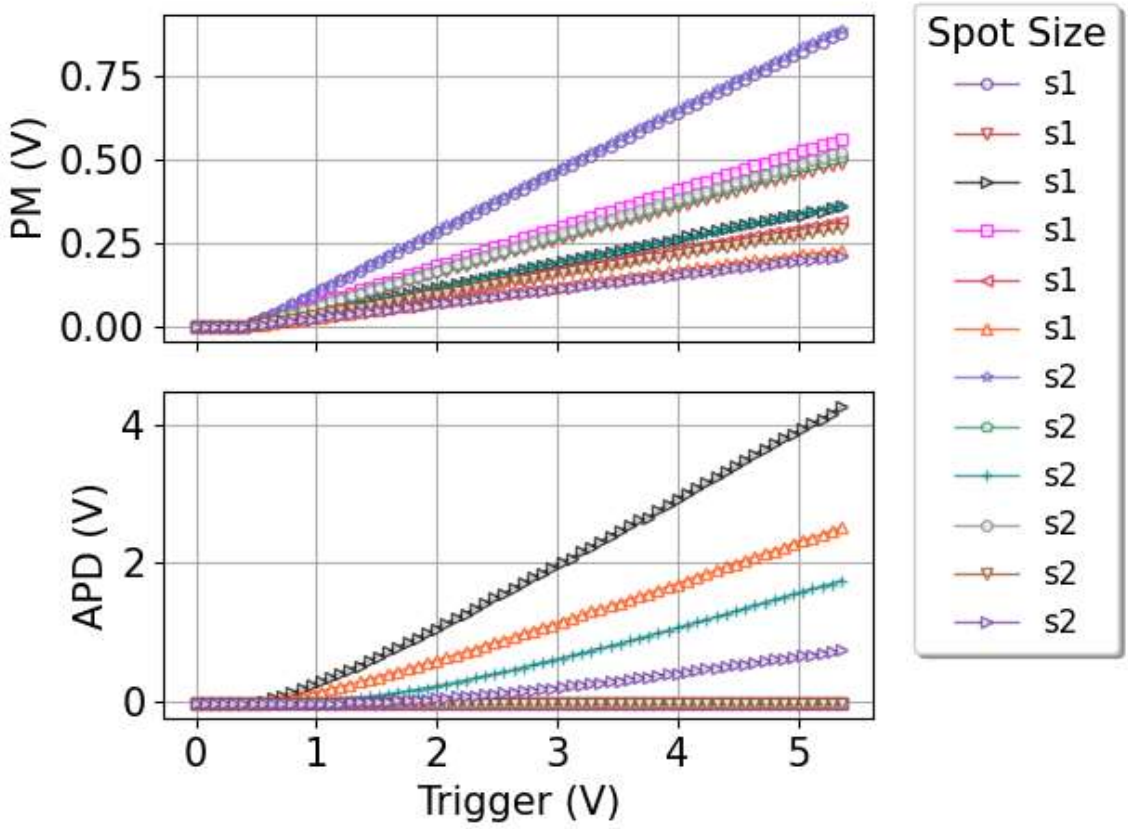
In [9]:

```
uData.get('steadydata', dict(fromPt=400, toPt=1000, precision=2), includeDiluter=True ,incl
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [10]:

```
fig, axs = uData.view(x='trigger', yList=['pm', 'apd'], includeDiluter=True, includeEmpty=T
```



Removing background

In [11]:

```
uData.get('removebackground', dict(channel='apd', baseline='trigger < 0.2'), includeEmpty=T
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

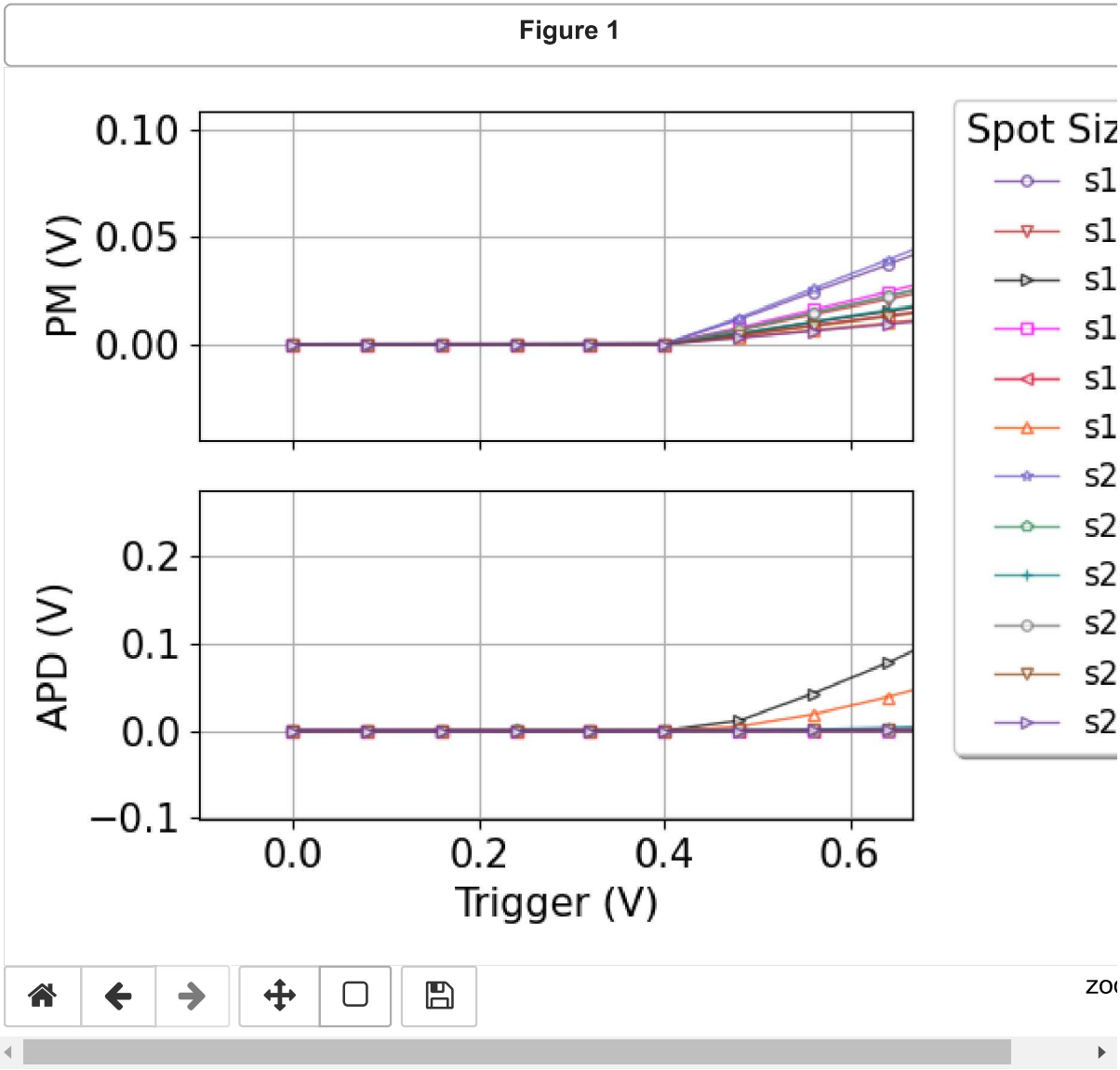
In [12]:

```
uData.get('removebackground', dict(channel='pm', baseline='trigger < 0.2'), includeEmpty=Tr
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [13]:

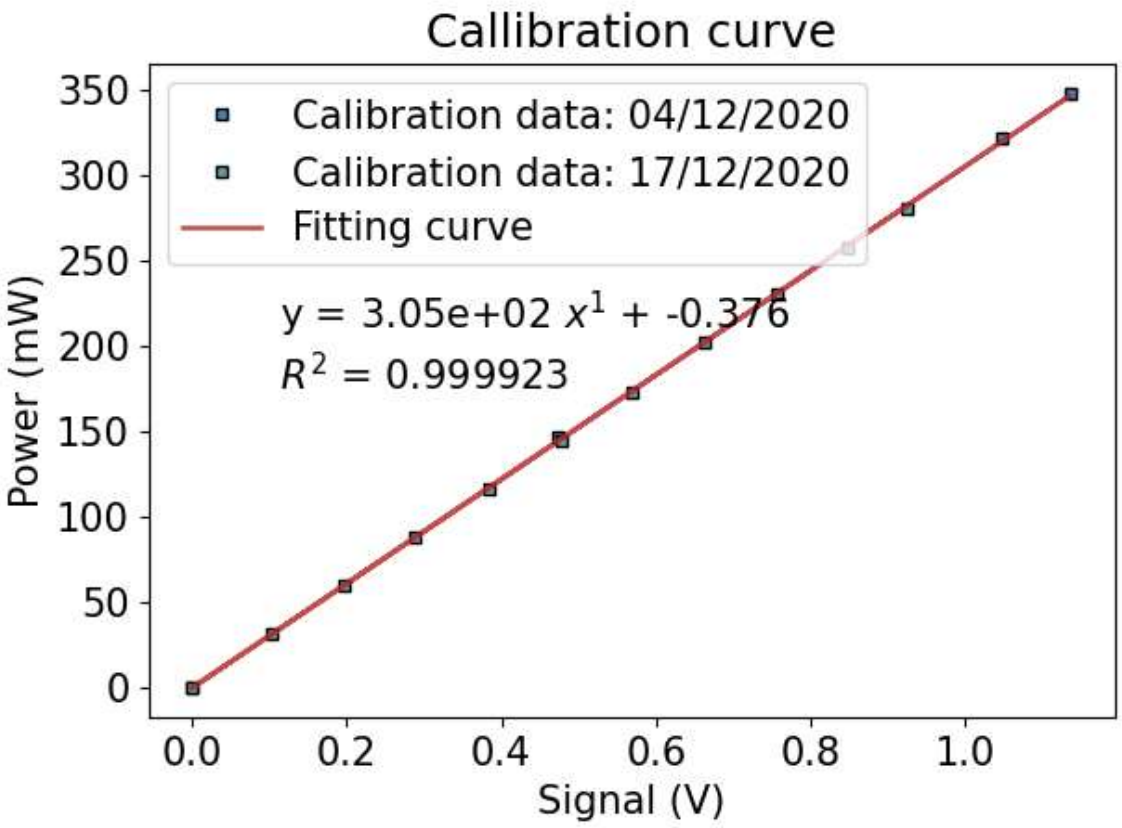
```
fig, axs = uData.view(x='trigger', yList=['pm', 'apd'], includeDiluter=True, includeEmpty=T
```



Power meter calibration

In [14]:

```
coeffs, r_val = calibratePM(laser=976, pmRange=600, degree=1)
```



In [15]:

```
coeffs
```

Out[15]:

```
[305.16892805223625, -0.37612596104816115]
```

In [17]:

```
pmCalibfunc600 = lambda x: coeffs[0] * x + coeffs[1]
```

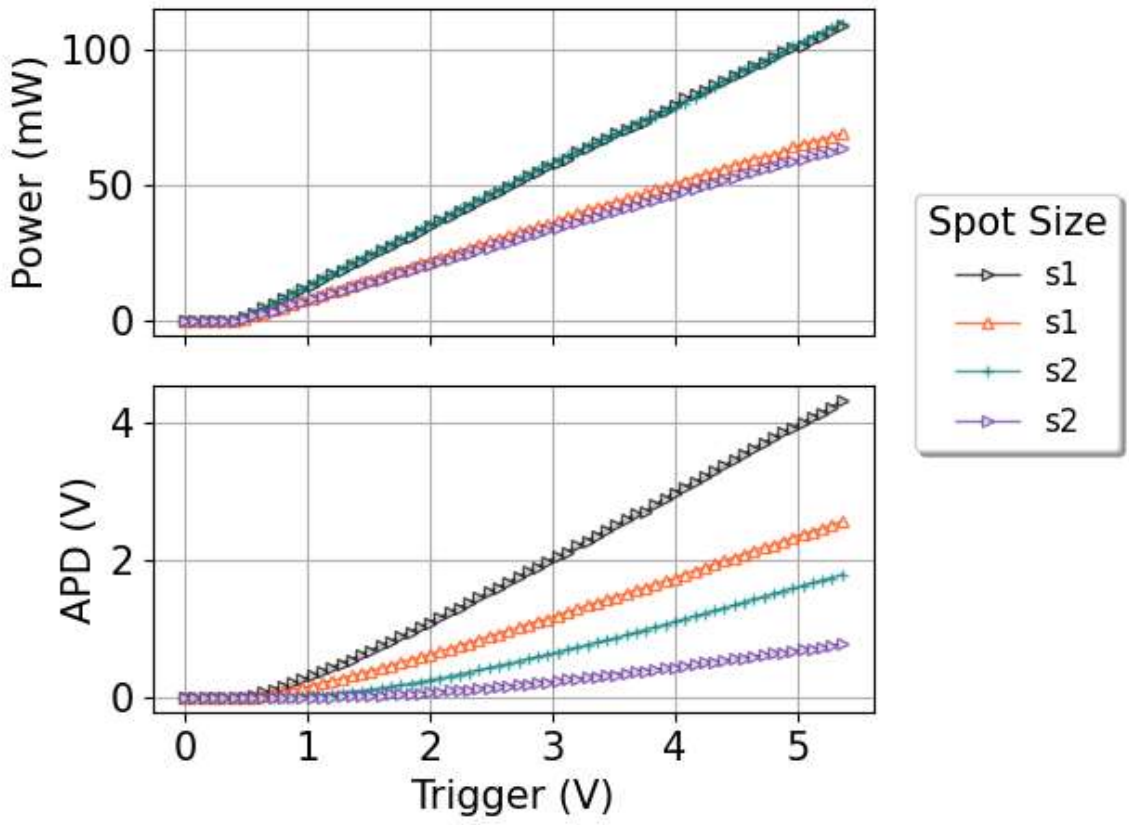
In [18]:

```
uData.get('calibrate', dict(calibFunction=pmCalibfunc600, channel='pm', recalculate=True),
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [19]:

```
fig, axs = uData.view(x='trigger', yList=['transmitted_power', 'apd'], label='spot_size')
```



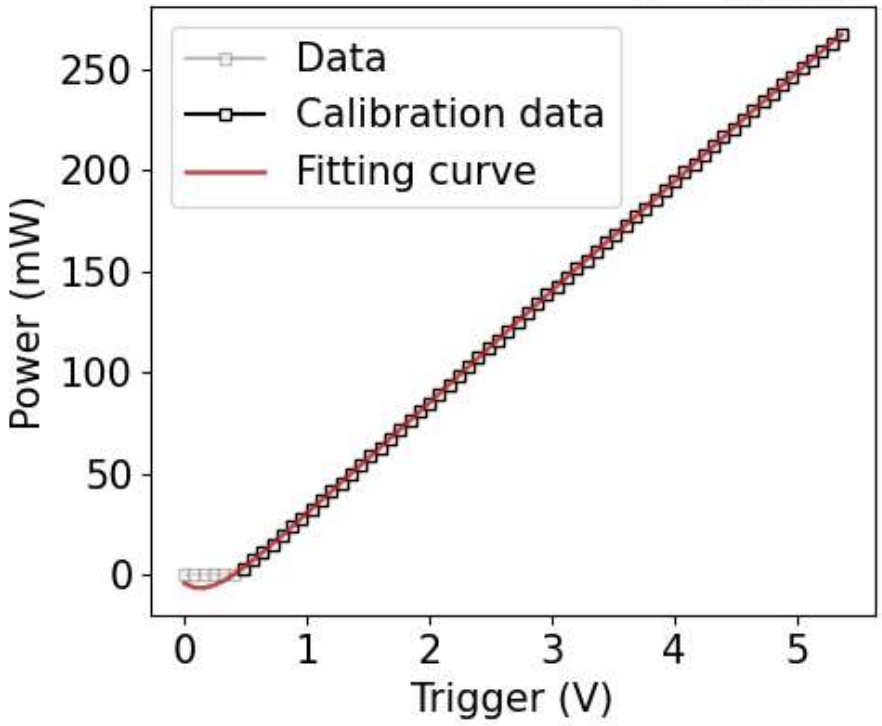
Trigger calibration

In [20]:

```
coeffs, r_val = uData.sample(0).polyfit(x='trigger', y='transmitted_power', degree=12, cond
triggerCalib = lambda x: np.poly1d(coeffs)(x)
```

$e+02\ x^5 + 2.51e+02\ x^4 + -2.77e+02\ x^3 + 2.1e+02\ x^2 + -42.0\ x^1 + -4.34$

$R^2 = 0.999962$



In [21]:

```
uData.get('calibrate', dict(calibFunction=triggerCalib, channel='trigger', recalculate=True
```

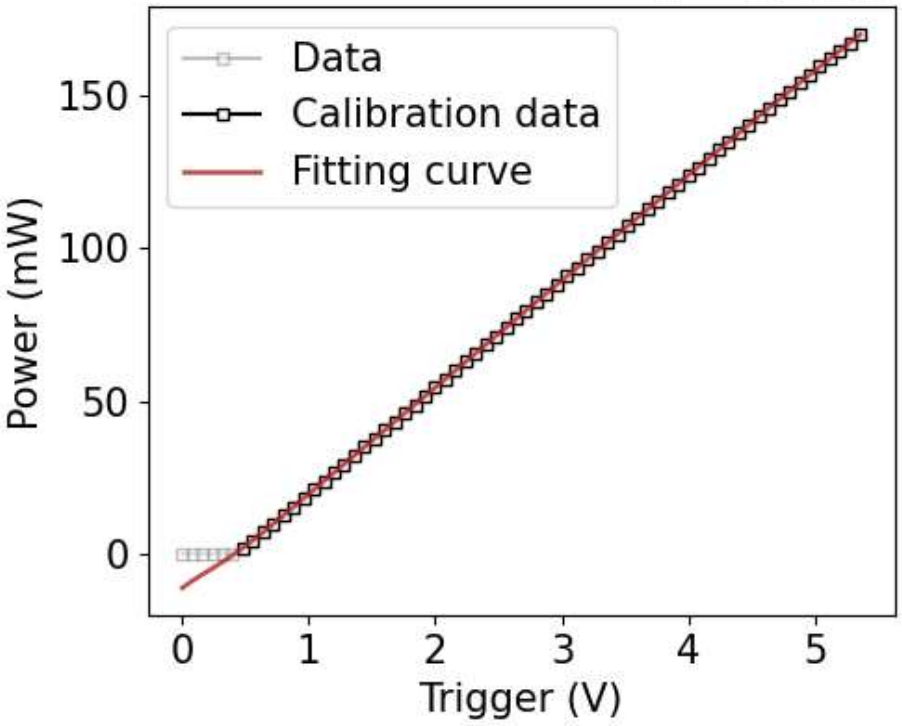
```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
```

In [22]:

```
coeffs, r_val = uData.sample(3).polyfit(x='trigger', y='transmitted_power', degree=12, cond
triggerCalib = lambda x: np.poly1d(coeffs)(x)
```

$$1.68e{+}02\,x^5 + -2.03e{+}02\,x^4 + 1.46e{+}02\,x^3 + -48.2\,x^2 + 32.2\,x^1 + -11.1$$

$$R^2 = 0.999972$$



In [23]:

```
uData.get('calibrate', dict(calibFunction=triggerCalib, channel='trigger', recalculate=True
```

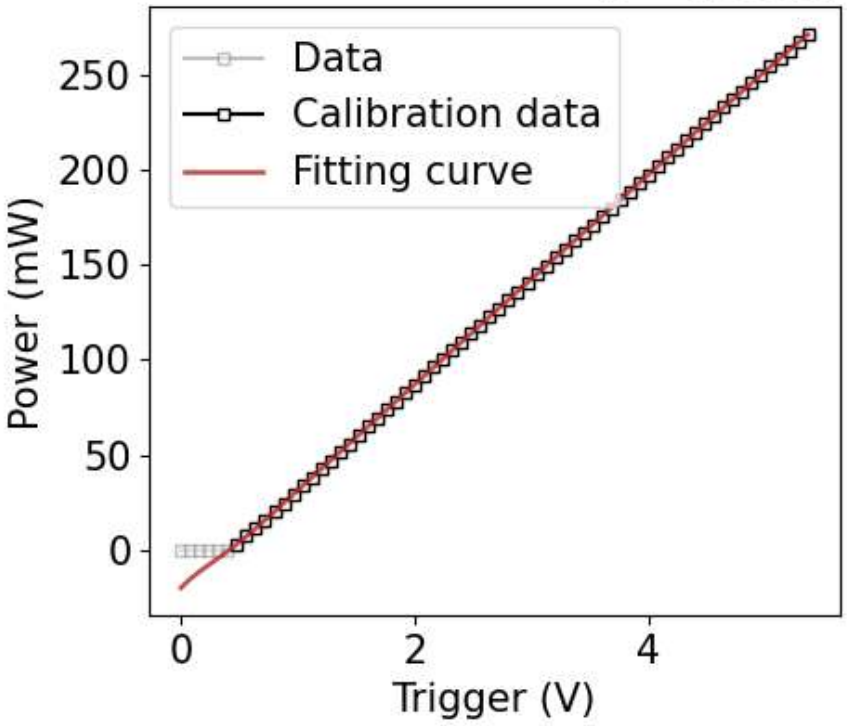
```
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
```

In [24]:

```
coeffs, r_val = uData.sample(6).polyfit(x='trigger', y='transmitted_power', degree=12, cond
triggerCalib = lambda x: np.poly1d(coeffs)(x)
```

$+02\ x^5 + -4.64e+02\ x^4 + 3.44e+02\ x^3 + -1.33e+02\ x^2 + 67.9\ x^1 + -19.7$

$R^2 = 0.999974$



In [25]:

```
uData.get('calibrate', dict(calibFunction=triggerCalib, channel='trigger', recalculate=True
```

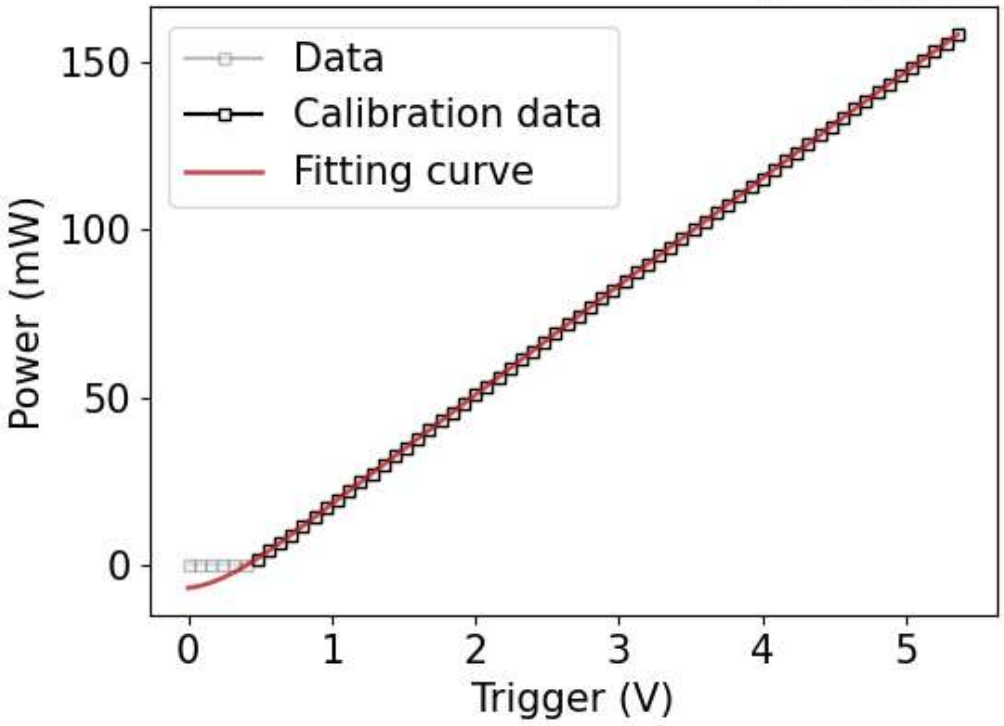
```
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
```

In [26]:

```
coeffs, r_val = uData.sample(9).polyfit(x='trigger', y='transmitted_power', degree=12, cond
triggerCalib = lambda x: np.poly1d(coeffs)(x)
```

$9\,x^7 + -9.9\,x^6 + 15.3\,x^5 + -7.13\,x^4 + -19.5\,x^3 + 39.0\,x^2 + 4.22\,x^1 + -7.01$

$R^2 = 0.999960$



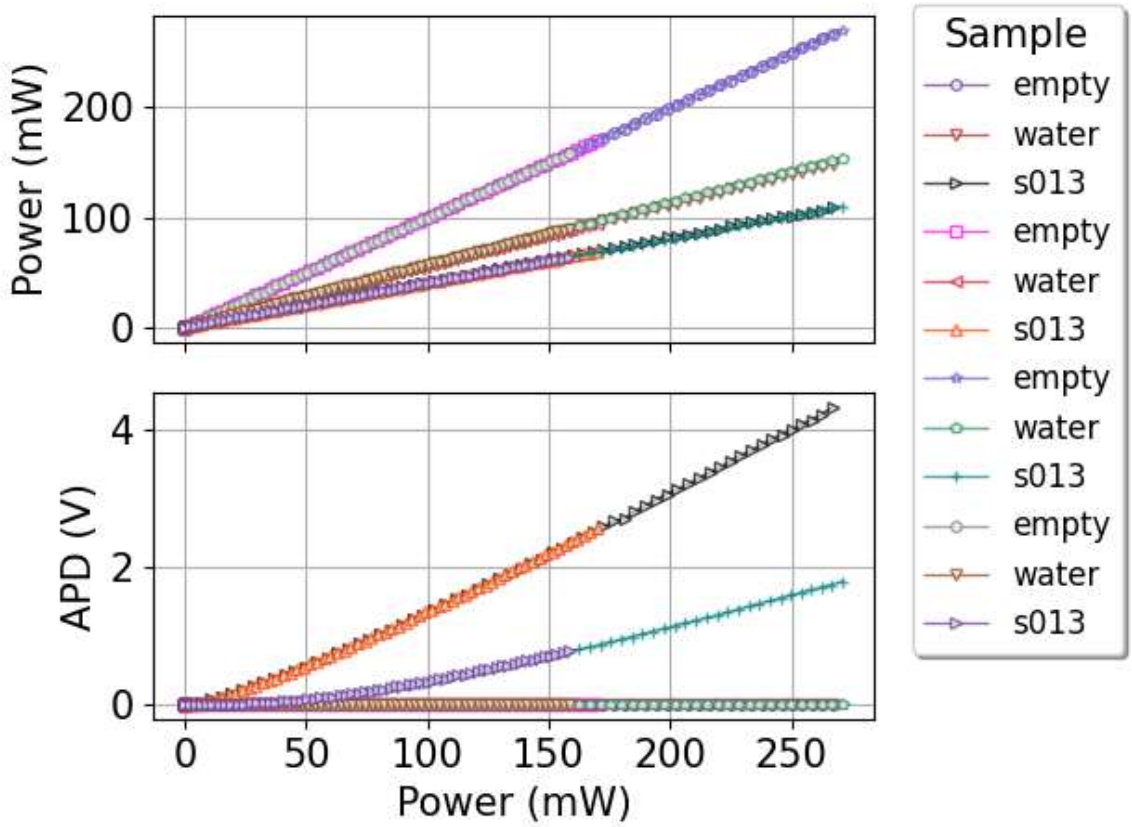
In [27]:

```
uData.get('calibrate', dict(calibFunction=triggerCalib, channel='trigger', recalculate=True
```

```
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [28]:

```
fig, axs = uData.view(x='laser_power', yList=['transmitted_power', 'apd'], label='sample',
```



Power density at the centre of the cuvette

In [24]:

```
help(Sample.powdensatcentre)
```

```
Help on function powdensatcentre in module lab:

powdensatcentre(self, beamWidth=None, recalculate=False)
    Calculates power density at the centre of the cuvette.
    Power at the centre/beam area.

    Args:
        recalculate (bool): if True it recalculates the power at the centre
of the cuvette.
```

In [177]:

```
help(Sample.poweratcentre)
```

```
Help on function poweratcentre in module lab:

poweratcentre(self, recalculate=False)
    Calculates power at the centre of the cuvette.
    Given by $\sqrt(P_o * P)$

    Args:
        recalculate (bool): if True it recalculates the power at the centre
of the cuvette.
```

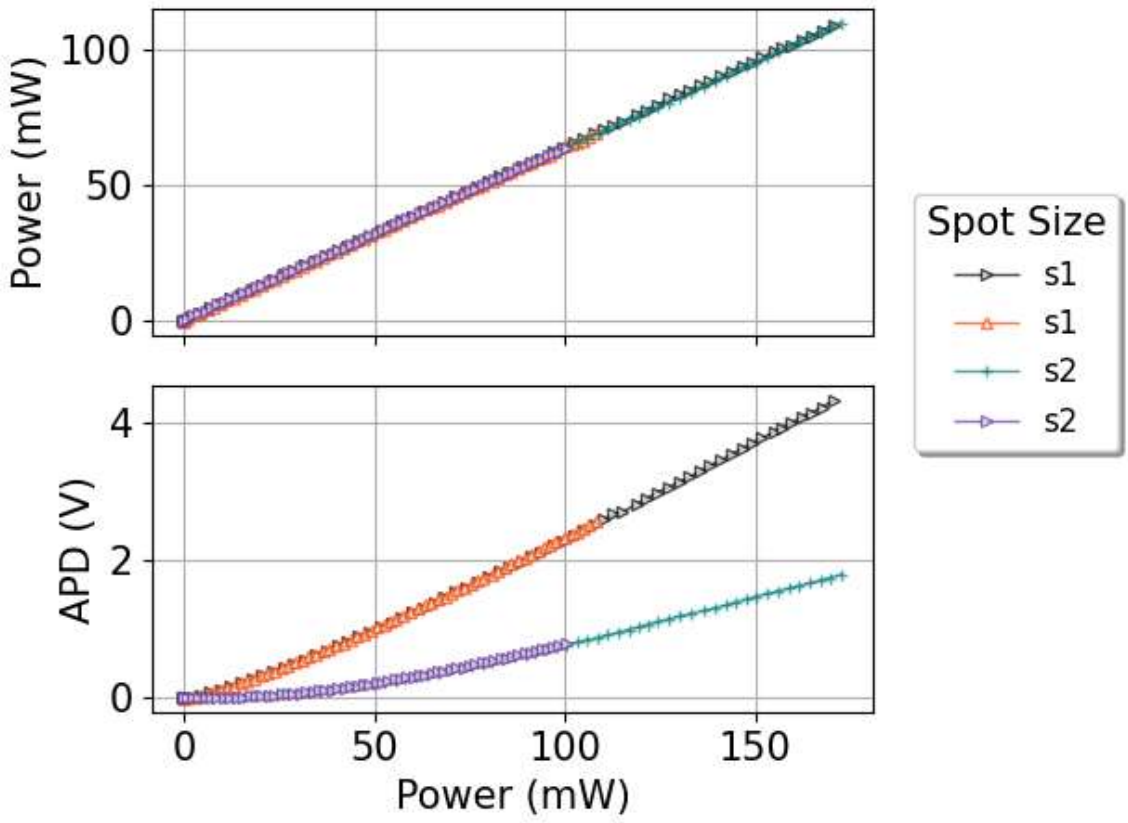
In [29]:

```
uData.get('powdensatcentre', dict(recalculate=True), includeDiluter=True, includeEmpty=True
```

```
20210430-120828: empty - Success!
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-115026: empty - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-112928: empty - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114220: empty - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [30]:

```
fig, axs = uData.view(x='power_at_centre', yList=['transmitted_power', 'apd'], label='spot_
```



APD calibration

In [31]:

```
# Coefficient obtained from Apd-calibration notebook
coeffs = [0.00525246, 0] # dye reabsorption included
apdCalibfuncMin = lambda x: np.poly1d(coeffs)(x)
```

In [32]:

```
uData.get('calibrate', dict(calibFunction=apdCalibfuncMin, channel='apd', recalculate=True)
```

```
20210430-120627: ucnp - Success!
20210430-120134: ucnp - Success!
20210430-113345: ucnp - Success!
20210430-114054: ucnp - Success!
```

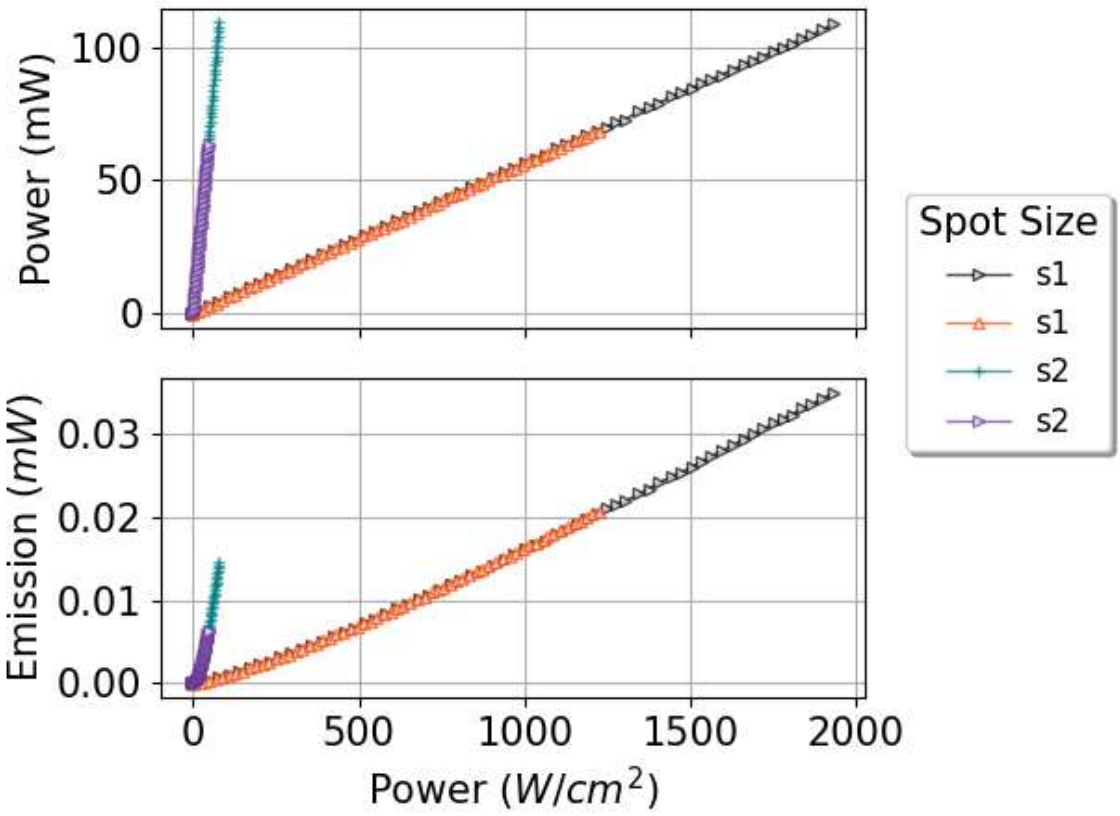
In [33]:

```
uData.get('emittedpower', dict(scatteringAtEmission=0.5, recalculate=True))
```

```
20210430-120627: ucnp - Success!
20210430-120134: ucnp - Success!
20210430-113345: ucnp - Success!
20210430-114054: ucnp - Success!
```

In [34]:

```
fig, axs = uData.view(x='pow_dens_at_centre', yList=['transmitted_power', 'emitted_power'],
```



Absorbance + Scattering

Sampleabsorbance represents the the absorption coeficient plus scattering.
Scattering is obtained with white light measurements.

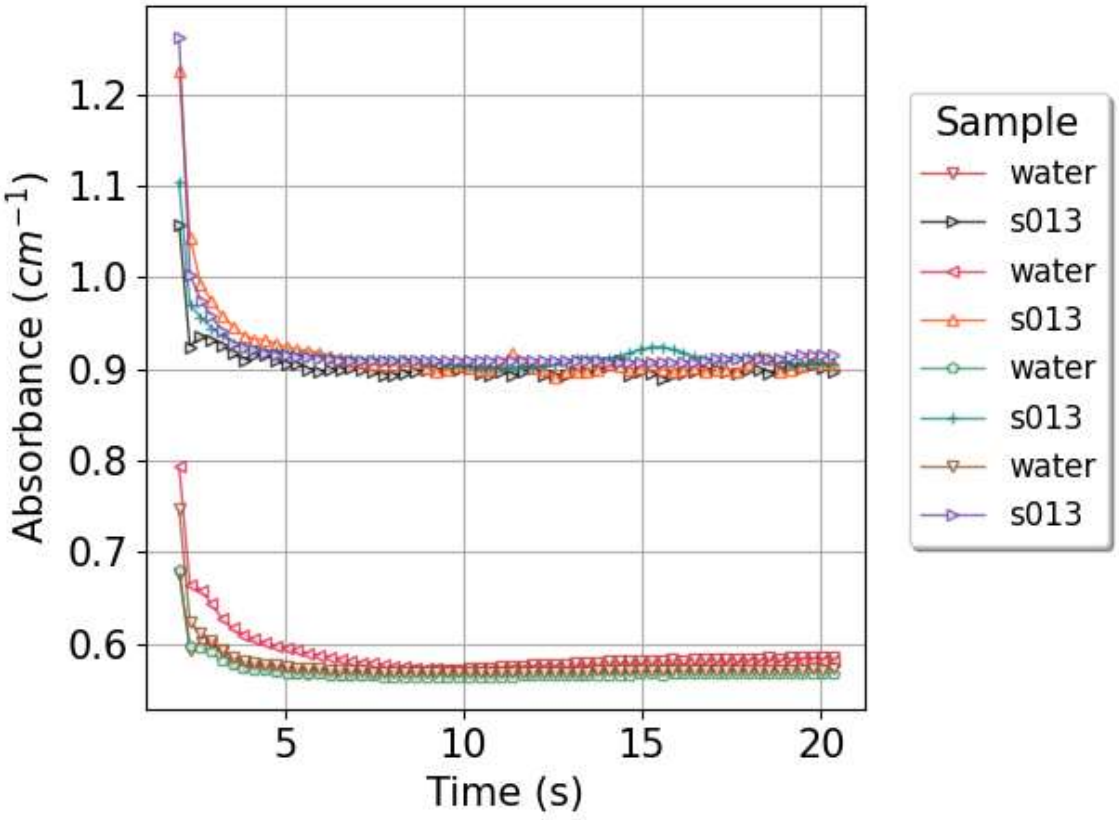
In [35]:

```
uData.get('sampleabsorbance',dict(recalculate=True), includeDiluter=True)
```

```
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [36]:

```
fig, axs = uData.view(x='time_daq', yList=['sample_absorbance'], label='sample', includeDil
```



In [37]:

```
for k in [1,4,7,10]:
    _, abMean, abStd, _ = uData.sample(k).sampleabsorbance(filterQuery='time_daq > 10', rec
    print(f'{uData.sample(k)._dataID}-{uData.sample(k)._sampleType}: {abMean=}, {abStd=}')
```

```
20210430-120923-diluter: abMean=0.5802187177682696, abStd=0.0039005442047185
99
20210430-120020-diluter: abMean=0.5750551746784278, abStd=0.0032505589472063
574
20210430-113227-diluter: abMean=0.5655113426351803, abStd=0.0013235354316370
31
20210430-114329-diluter: abMean=0.5702465562633373, abStd=0.0010001993915492
037
```

In [38]:

```
uData.get('absorbance', dict(recalculate=True), includeDiluter=True)
```

```
20210430-120923: diluter - Success!
20210430-120627: ucnp - Success!
20210430-120020: diluter - Success!
20210430-120134: ucnp - Success!
20210430-113227: diluter - Success!
20210430-113345: ucnp - Success!
20210430-114329: diluter - Success!
20210430-114054: ucnp - Success!
```

In [31]:

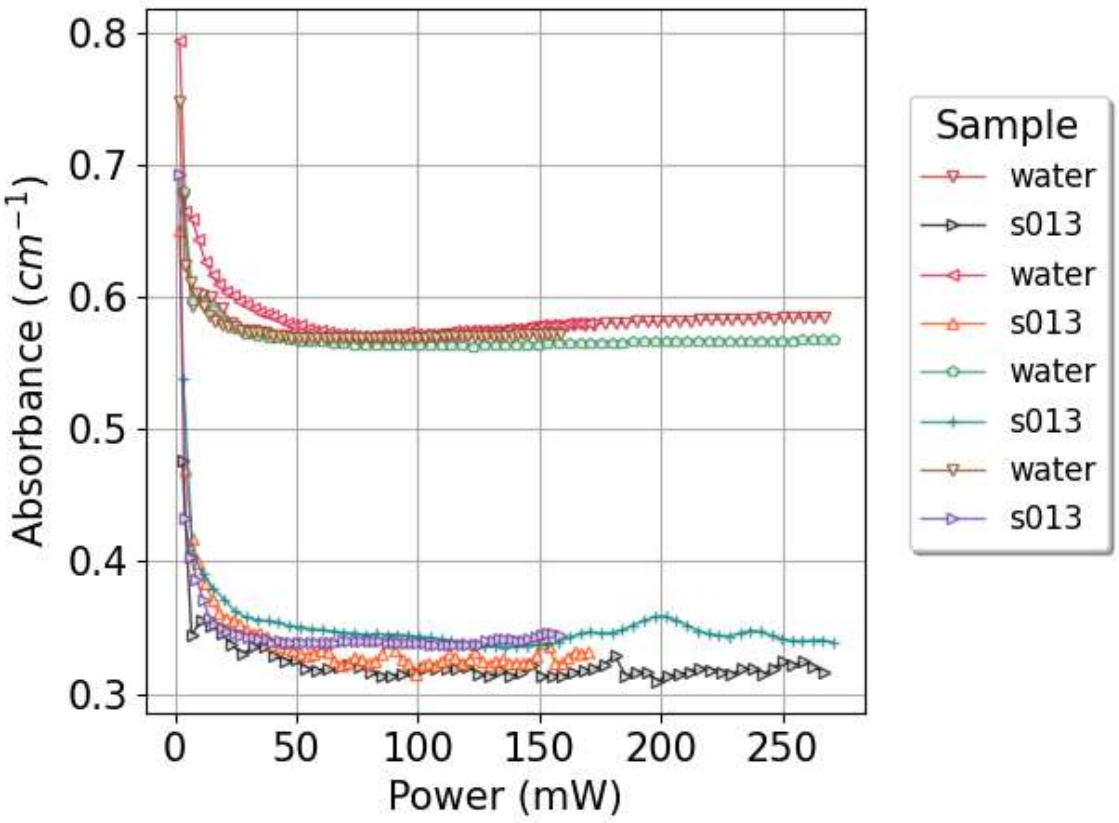
```
uData.sample(11).absorbance(filterQuery='time_daq > 10', recalculate=True)
```

Out[31]:

```
(0          NaN
 1          NaN
 2          NaN
 3          NaN
 4          NaN
          ...
 63    0.343571
 64    0.344981
 65    0.346035
 66    0.345985
 67    0.344327
 Name: absorbance, Length: 68, dtype: float64,
 0.34002209804944966,
 0.002499419719701178,
 '20210430-114054: ucnp - Success!')
```

In [39]:

```
fig, axs = uData.view(x='laser_power', yList=['absorbance'], label='sample', includeDiluter
```



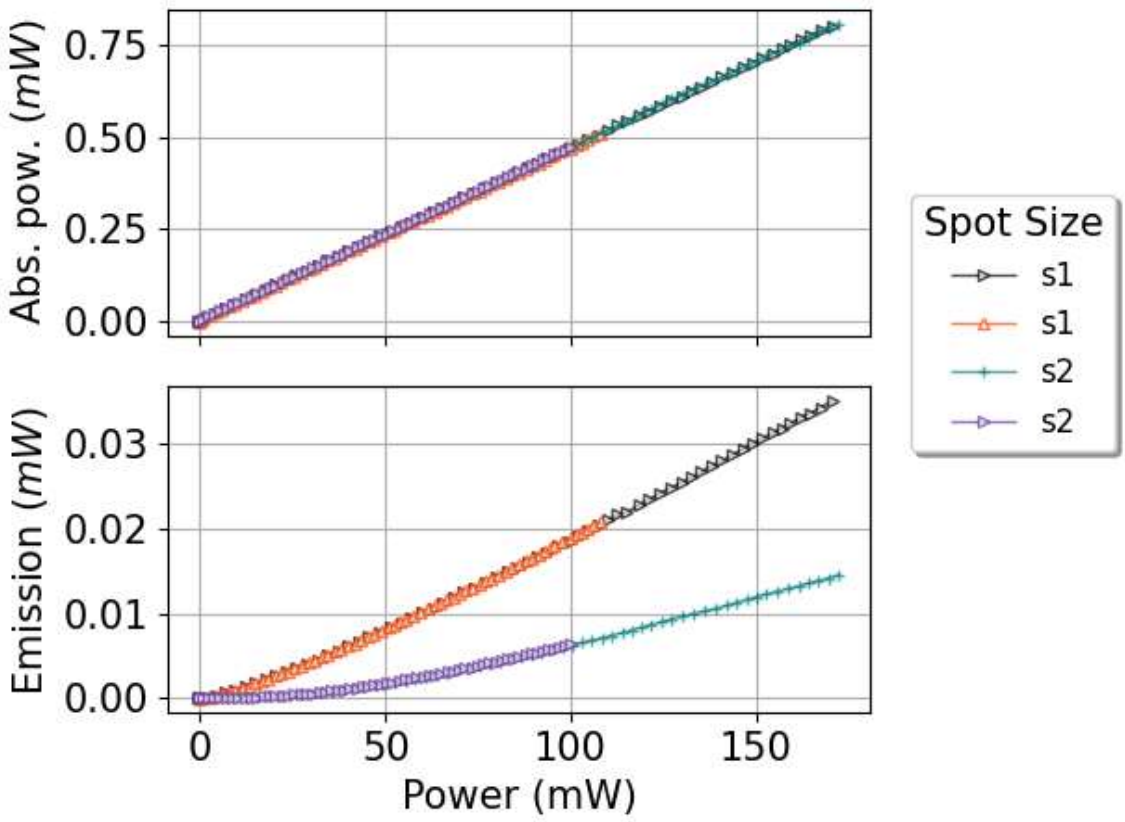
In [40]:

```
uData.get('absorbedpower', dict(filterQuery='time_daq > 10', recalculate=True, absorptionCo
```

```
20210430-120627: ucnp - Success!
20210430-120134: ucnp - Success!
20210430-113345: ucnp - Success!
20210430-114054: ucnp - Success!
```

In [41]:

```
fig, axs = uData.view(x='power_at_centre', yList=['absorbed_power', 'emitted_power'], label
```



Get kappa κ

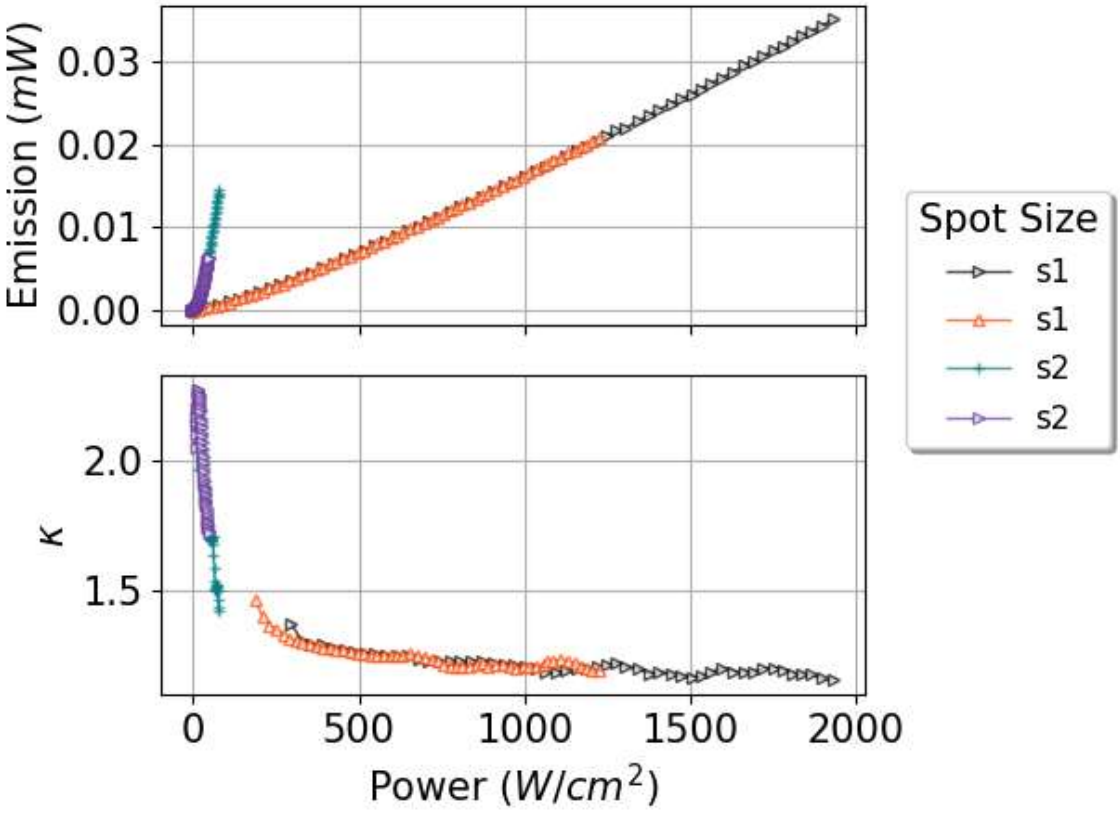
In [42]:

```
uData.get('kappa', dict(period=10, recalculate=True))
```

```
20210430-120627: ucnp - Success!
20210430-120134: ucnp - Success!
20210430-113345: ucnp - Success!
20210430-114054: ucnp - Success!
```

In [43]:

```
fig, axs = uData.view(x='pow_dens_at_centre', yList=['emitted_power', 'kappa'], label='spot
```



Quantum Yield ϕ

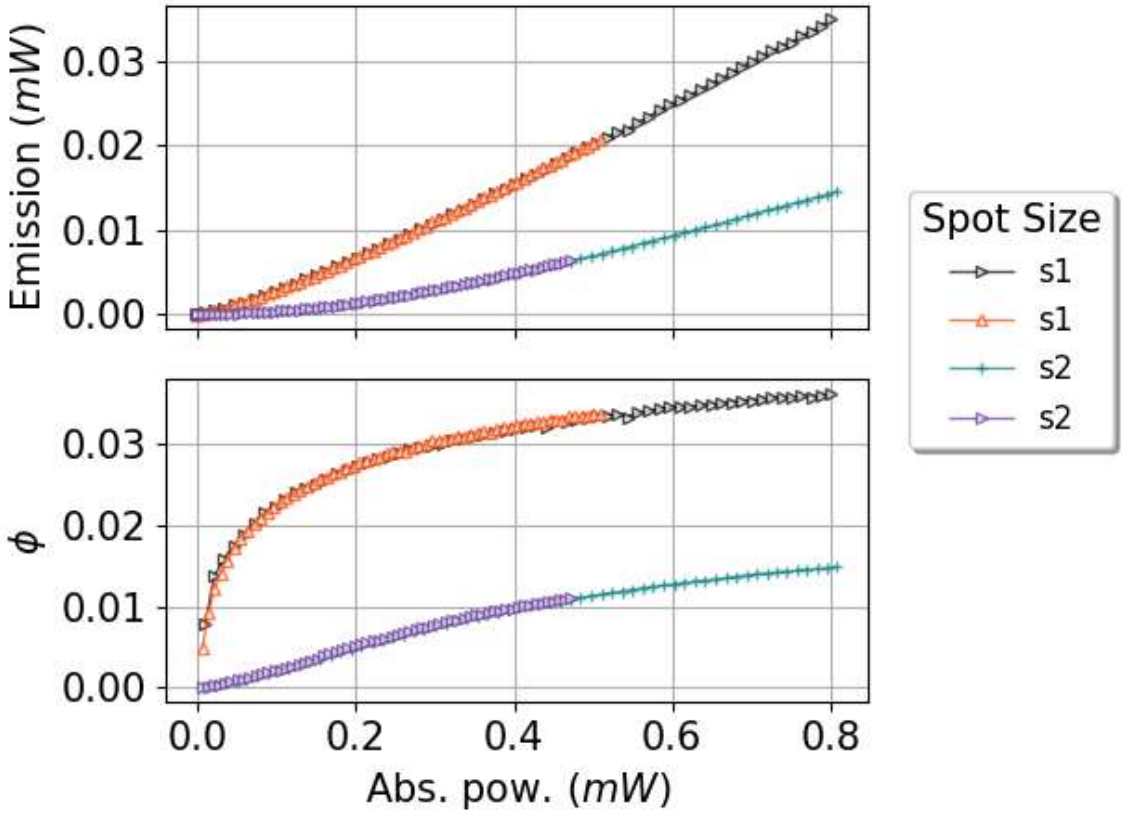
In [44]:

```
uData.get('quantumyield', dict(recalculate=True))
```

```
20210430-120627: ucnp - Success!
20210430-120134: ucnp - Success!
20210430-113345: ucnp - Success!
20210430-114054: ucnp - Success!
```

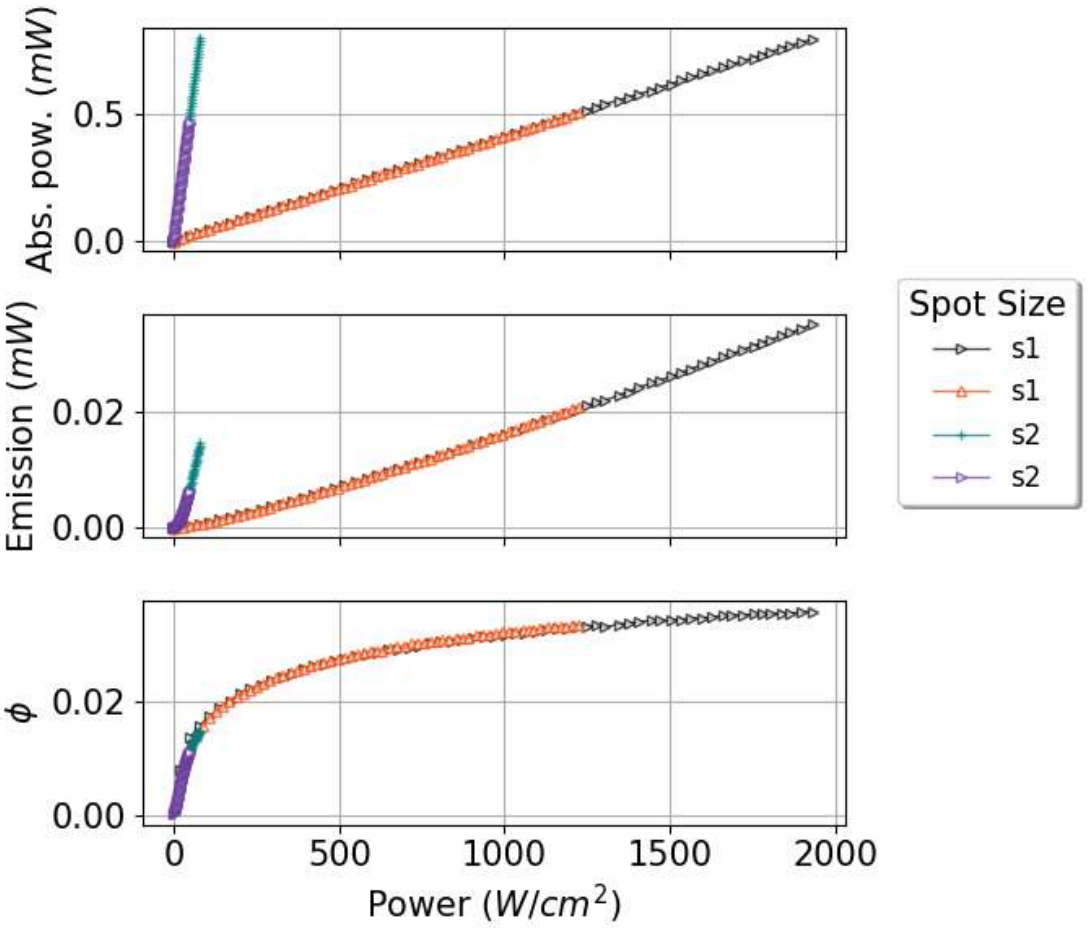
In [45]:

```
fig, axs = uData.view(x='absorbed_power', yList=['emitted_power', 'quantum_yield'], which=[
```



In [46]:

```
fig, axs = uData.view(x='pow_dens_at_centre', yList=['absorbed_power', 'emitted_power', 'qu
```



In [40]:

```
pl.tight_layout()
```

Fitting emission according to a simplified model

$$E = 2\phi_b \alpha_{976} A_{px} \frac{\lambda_e}{\lambda_a} \sum_k^N \frac{\rho_k^2}{\rho_b + \rho_k}$$

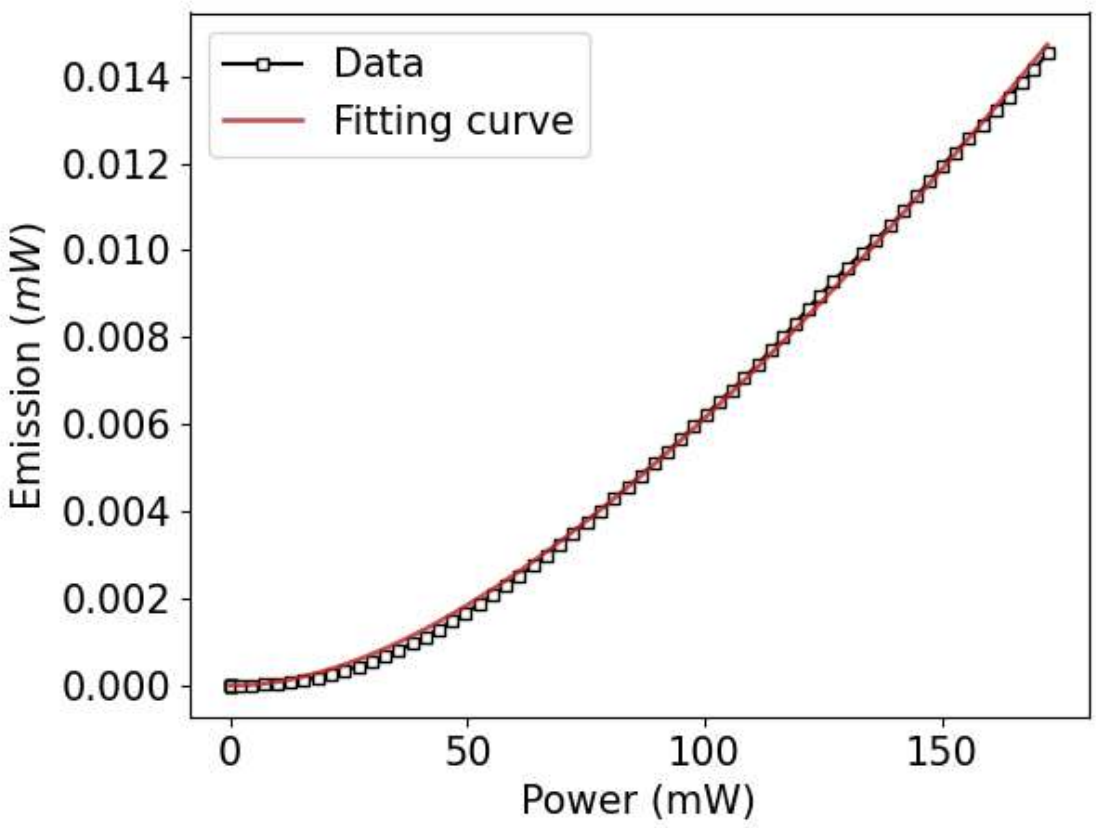
Being α_{976} the absorption coeficient of the UCNP's at 976 nm, A_{px} the area of a pixel, ρ power density

In [47]:

```
def formatcoefs(popt, pcov):
    perr = np.sqrt(np.diag(pcov))
    phib = popt[0]; phib_err = perr[0]
    rhob = popt[1]; rhob_err = perr[1]
    print(f'phi_b = {popt[0]:.4f}({perr[0]:.4f}), rho_b = {popt[1]/1000:.2f}({perr[1]/1000:
    return(phib, phib_err, rhob, rhob_err)
```

In [67]:

```
popt, pcov, emissionF8, _ = uData.sample(8).fitemission(bounds=([0, 0], [1, 1e+5]), show=Tr
phib, phib_err, rhob, rhob_err = formatcoefs(popt, pcov)
```




```
phi_b = 0.0189(0.0004), rho_b = 38.22(1.50)W/cm^2
```

Converting units

In [69]:

```
rhob /= 1000
rhob_err /= 1000
phib *= 100
phib_err *= 100
```

In [77]:

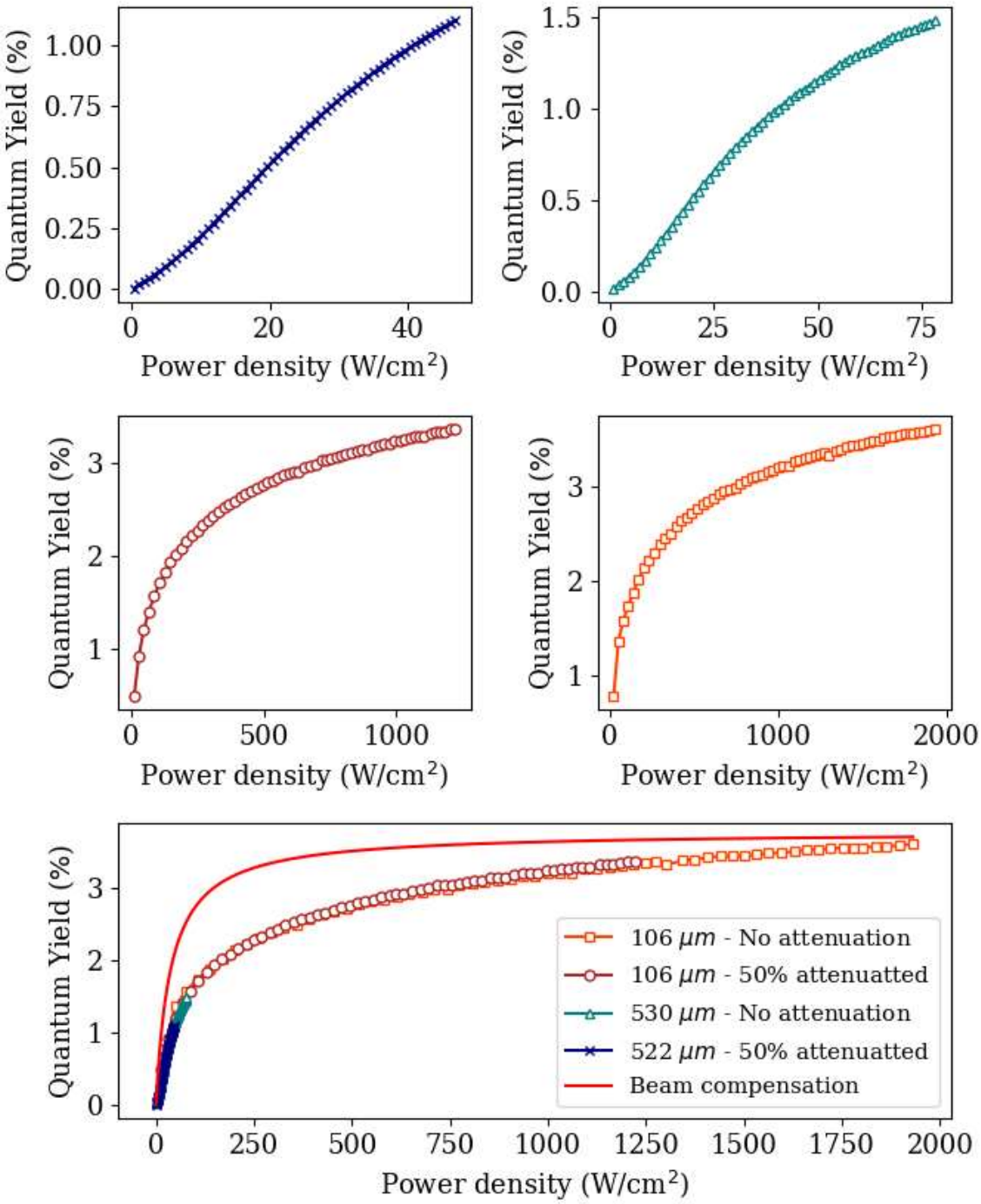
```
print(f"{rhob=:.1f} pm {rhob_err=:.1f}, {phib=:.2f} pm {phib_err:.2f}")
```

```
rhob=38.2 pm rhob_err=1.5, phib=1.89 pm 0.04
```

Fitting graphs

In [81]:

```
pl.rcParams.update({'font.size': 13})
pl.rcParams.update({'font.family':'serif'})
def qy(p, phib, rhob): return (phib * p / (rhob + p))

fig = pl.figure(figsize=(7,8.5))
xlabel = 'Power density (W/cm$^2$)'
ylabel = 'Quantum Yield ($\%$)'

ax1 = pl.subplot2grid((3,2), (0,0))
ax2 = pl.subplot2grid((3,2), (0,1))
ax3 = pl.subplot2grid((3,2), (1,0))
ax4 = pl.subplot2grid((3,2), (1,1))
ax5 = pl.subplot2grid((3,2), (2,0), colspan=2)

ax1.set_xlabel(xlabel)
ax2.set_xlabel(xlabel)
ax3.set_xlabel(xlabel)
ax4.set_xlabel(xlabel)
ax5.set_xlabel(xlabel)
ax1.set_ylabel(ylabel)
ax2.set_ylabel(ylabel)
ax3.set_ylabel(ylabel)
ax4.set_ylabel(ylabel)
ax5.set_ylabel(ylabel)

# fig, ax = pl.subplots()
ax4.plot(uData.sample(2).data()['pow_dens_at_centre'], uData.sample(2).data()['quantum_yiel
ax3.plot(uData.sample(5).data()['pow_dens_at_centre'], uData.sample(5).data()['quantum_yiel
ax2.plot(uData.sample(8).data()['pow_dens_at_centre'], uData.sample(8).data()['quantum_yiel
ax1.plot(uData.sample(11).data()['pow_dens_at_centre'], uData.sample(11).data()['quantum_yi


ax5.plot(uData.sample(2).data()['pow_dens_at_centre'], uData.sample(2).data()['quantum_yiel
ax5.plot(uData.sample(5).data()['pow_dens_at_centre'], uData.sample(5).data()['quantum_yiel
ax5.plot(uData.sample(8).data()['pow_dens_at_centre'], uData.sample(8).data()['quantum_yiel
ax5.plot(uData.sample(11).data()['pow_dens_at_centre'], uData.sample(11).data()['quantum_yi

# fitted model
power = np.arange(0, 1932, 0.5)
ax5.plot(power, qy(power, 2*phib, rhob), '-', color='red', label='Beam compensation')
pl.ylabel(ylabel)
pl.xlabel(xlabel)
pl.legend(fontsize=11)
fig.tight_layout()
```



In [82]:

```
# save figure
fig.savefig('../plots/Quantum_Yield_pannel.png')
```

In [58]:

```
# emission
data = '../data/emission/s013_20210521_emission_QEP025961.txt'
data = pd.read_csv(data, sep='\t', skiprows=15, names=['Wavelength', 'Intensity'])
```

In [59]:

```
data
```

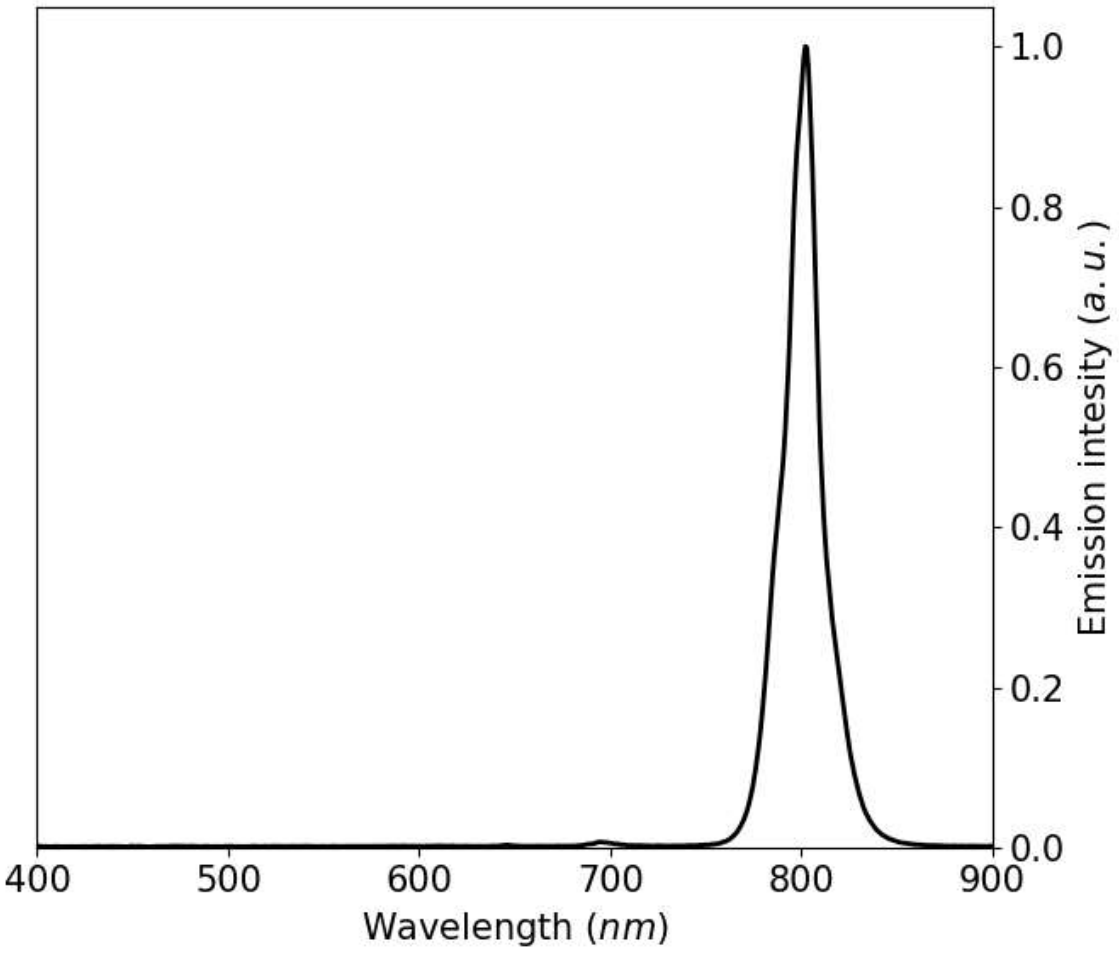
Out[59]:

	Wavelength	Intensity
0	346.573	1.25
1	347.359	-0.75
2	348.146	-1.75
3	348.933	246.25
4	349.719	227.25
...	...	...
1038	1123.340	397.25
1039	1124.047	-3.75
1040	1124.754	-5.75
1041	1125.460	-2.75
1042	1126.167	-0.75

1043 rows × 2 columns

In [60]:

```
pl.rcParams.update({'font.size': 15})
fig, ax = pl.subplots(figsize=(7, 6))
ax.plot(data['Wavelength'], data['Intensity']/data['Intensity'].max(), color='k', label='Em

ax.set_xlabel('Wavelength ($nm$)')
ax.set_ylabel('Emission intesity ($a.u.$)')
ax.set_ylim([0,1.05])
ax.set_xlim([400,900])
ax.yaxis.tick_right()
ax.yaxis.set_label_position("right")
# ax.grid()
pl.tight_layout()
```



In [50]:

```
fig.savefig('../plots/spectrum.pdf')
```

Exporting analysed data that generate the plots

In [83]:

```
for idx in uData.details().index:
    fileName = '../data/analysed-data/' + uData.details().loc[idx, 'exp_id'] + '_' + uData.
    metadataName = '../data/analysed-data/' + uData.details().loc[idx, 'exp_id'] + '_metada

    uData.sample(idx).data().to_csv(fileName)
    uData.sample(idx)._metadata.to_csv(metadataName)
```

In [84]:

```
uData.sample(11)._metadata
```

Out[84]:

	field_name	data_type	data_format	example	standard_units	plot_label	des
0	time	float	^([0-9]?\.|[1-9][\d]*\.)[0-9]*$	123423.982734	seconds	Time (s)	nu
1	trigger	float	^[-]?([0-9]?\.|[1-9][\d]*\.)[0-9]*$	-1.998340	volts	Trigger (V)	D
2	pm	float	^[-]?([0-9]?\.|[1-9][\d]*\.)[0-9]*$	-1.998340	volts	PM (V)	D
3	apd	float	^[-]?([0-9]?\.|[1-9][\d]*\.)[0-9]*$	-1.998340	volts	APD (V)	D
4	time_daq	float	^[-]?([0-9]?\.|[1-9][\d]*\.)[0-9]*$	0.029380	s	Time (s)	Dac
5	transmitted_power	float	^[0-9]*.[0-9]*	5.500000	mW	Power (mW)	Tra (ca
6	laser_power	float	^[0-9]*.[0-9]*	5.500000	mW	Power (mW)	(fro ca
7	power_at_centre	float	^[0-9]*.[0-9]*	100.565000	mW	Power (mW)	I th
8	pow_dens_at_centre	float	^[0-9]*.[0-9]*	10000.565000	W/cm^2	Power (W/cm^2)	d th
9	apd_power	float	^[0-9]*.[0-9]*	5.500000	mW	Power (mW)	Ap - (f ca
10	emitted_power	float	^[0-9]*.[0-9]*	10.565000	mW	Emission (mW)	po di
11	sample_absorbance	float	^0.[0-9]*	0.400000	(cm^{-1})	Absorbance (cm^{-1})	Abs obt log wl

	field_name	data_type	data_format	example	standard_units	plot_label	des
12	absorbance	float	^0.[0-9]*	0.400000	cm^{-1}	Absorbance (cm^{-1})	Abs obt abs
13	absorbed_power	float	^0.[0-9]*	0.400000	mW	Abs. pow. (mW)	A ca
14	kappa	float	^[0-9]*.[0-9]*	10000.565000	-	κ	slo log
15	quantum_yield	float	^[0-9]*.[0-9]*	10000.565000	-	ϕ	y lam la
16	power_density	float	^[0-9]*.[0-9]*	5.500000	W/cm^2	Power (W/cm^2)	fro
17	absorption	float	^[0-9]*.[0-9]*	5.500000	mW/cm^2	Absorption (mW/cm^3)	Abs A p vol
18	emission	float	^[0-9]*.[0-9]*	5.500000	mW/cm^2	Emission (mW/cm^3)	Er p vol

In [ ]: