PhytoPhotoUtils

Release 01-10-2019

Apr 12, 2021

Getting started

1 Installation **1**

2 Loading **3**

3 Saturation **7**

4 Relaxation **11**

5 General Tools **15**

6 Fluorescence Light Curves **17**

7 Spectral Correction **21**

8 Plotting **25**

9 API Documentation **27**
- 9.1 Loading . . . 27
- 9.2 General Tools . . . 31
- 9.3 Saturation . . . 32
- 9.4 Relaxation . . . 34
- 9.5 Spectral Correction . . . 37
- 9.6 Fluorescence Light Curves . . . 39
- 9.7 Plotting . . . 42

10 Citing and Referencing **45**
- 10.1 Reference . . . 45
- 10.2 Authors and contributors . . . 45

11 Change Log **47**

12 Contributing **51**
- 12.1 Pull Request Process . . . 51
- 12.2 Code of Conduct . . . 51

13 Indices and tables **53**

Python Module Index **55**

Index 57

CHAPTER 1

Installation

Download and install with the latest updates:

```
$ git clone git@gitlab.com:tjryankeogh/phytophotoutils.git
$ cd PhytoPhotoUtils
$ python setup.py install
```

CHAPTER 2

Loading

phyto_photo_utils._load.**load_FASTTrackaI_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=120*, *irrad=None*)

Process the raw data file and convert to a csv with standard formatting.

Parameters

- **file** (*str*) – The path directory to the .000 data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*dir, default=None*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=120*) – The number of flashlets in the protocol.
- **irrad** (*int, default=None*) – The light/dark chamber photons per count from the calibration file.

Returns

- **df** (*pandas.DataFrame, shape=[n,8]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.

- **channel** (*np.array, dtype=str, shape=[n,]*) – The chamber used for measurements, A = light chamber, B = dark chamber.
- **gain** (*np.array, dtype=int, shape=[n,]*) – The gain settings of the instrument.

Example

```
>>> fname = './data/raw/instrument/fasttrackai/FASTTrackaI_example.csv'
>>> output = './data/raw/ppu/fasttrackai/'
>>> df = ppu.load_FASTTrackaI_files(fname, append=False, save_files=True, res_
↪path=output, seq_len=120, irrad=545.62e10)
```

phyto_photo_utils._load.**load_FIRe_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=160*, *gain_value=None*, *flen=1e-06*, *irrad=None*, *continuous=False*, *light_step=False*, *single_turnover=True*)

Process the raw data file (.000 format) and convert to a csv with standard formatting.

Parameters

- **file** (*str*) – The path directory to the data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*str, default=None*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int , default=160*) – The number of flashlets in the protocol. Only required if continuous = True.
- **gain** (*int, default=None*) – The gain value set when performing measurements. Only required if continous = True. Discrete Gain values are recorded in raw data files.
- **flen** (*float, default=1e-6*) – The flashlet length in seconds.
- **irrad** (*int, default=None*) – The LED output in μE m^2 s^{-1}. Only required if continuous = True.
- **continuous** (*bool, default=False*) – If True, will load files from the continuous format. If False, will load the discrete file format.
- **light_step** (*bool, default=False*) – If True, will load files from a discrete format FLC file. If False, will load the discrete file format with no light steps.
- **single_turnover** (*bool, default=True*) – If True, will load the saturation and relaxation from the single turnover measurement. If False, will load the multiple turnover measurement.

Returns

- **df** (*pandas.DataFrame, shape=[n,6]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.

- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.

Example

```
>>> fname = './data/raw/instrument/fire/FIRe_example.000'
>>> output = './data/raw/ppu/fire/'
>>> df = ppu.load_FIRe_files(fname, append=False, save_files=True, res_
↪path=output, seq_len=160, flen=1e-6, irrad=47248)
```

phyto_photo_utils._load.**load_FastOcean_files**(*file_*, *append=False*, *save_files=False*, *led_separate=False*, *res_path=None*, *seq_len=125*, *seq_reps=None*, *flen=1e-06*, *delimiter=','*, *FastAct1=True*, *Single_Acq=False*)

Process the raw data file and convert to a csv with standard formatting.

Parameters

- **file** (*dir*) – The path directory to the .csv data file from the FastOcean with either the FastAct11 or FastAct12 laboratory system.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together. Not applicable if Single_Acq = True.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **led_separate** (*bool, default=False*) – If True, the protocols will be separated dependent upon the LED sequence.
- **res_path** (*dir*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=125*) – The number of flashlets in the protocol.
- **seq_reps** (*int, default=None*) – The number of sequences per acquisition or in FastAct2 multiple acquisition files the number of light steps.
- **flen** (*float, default=1e-6*) – The flashlet length in seconds.
- **delimiter** (*str, default=','*) – Specify the delimiter to be used by Pandas.read_csv for loading the raw files.
- **FastAct1** (*bool, default=True*) – If True, will load data from FastAct1 laboratory system format. If False, will load data from FastAct2 laboratory system format.
- **Single_Acq** (*bool, default=False*) – If True, will load a single acquisition data from either the FastAct1 or FastAct2 laboratory system, dependent upon whether FastAct1 is True of False.

Returns

- **df** (*pandas.DataFrame, shape=[n,7]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.

- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.
- **led_sequence** (*np.array, dtype=int, shape=[n,]*) – The LED combination using during the measurement, see example below.

Example

```
>>> fname = './data/raw/instrument/fire/FastOcean_example.csv'
>>> output = './data/raw/ppu/fastocean/'
>>> df = ppu.load_FastOcean_files(fname, append=False, save_files=True, led_
↪separate=False, res_path=output, seq_len=125, flen=1e-6)
>>> led_sequence == 1, LED 450 nm
>>> led_sequence == 2, LED 450 nm + LED 530 nm
>>> led_sequence == 3, LED 450 nm + LED 624 nm
>>> led_sequence == 4, LED 450 nm + LED 530 nm + LED 624 nm
>>> led_sequence == 5, LED 530 nm + LED 624 nm
>>> led_sequence == 6, LED 530 nm
>>> led_sequence == 7, LED 624 nm
```

phyto_photo_utils._load.**load_LIFT_FRR_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=228*)

Process the raw data file from the Soliense LIFT FRR and convert to a csv with standard formatting.

Parameters

- **file** (*dir*) – The path directory to the .000 data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*dir*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=228*) – The number of flashlets in the protocol.

Returns

- **df** (*pandas.DataFrame, shape=[n,7]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to seq_len
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.

CHAPTER 3

Saturation

phyto_photo_utils._saturation.**fit_saturation**(*pfd, flevel, seq, datetime, blank=0, sat_len=100, skip=0, ro=0.3, no_ro=False, calc_ro=True, fixed_ro=False, bounds=True, sig_lims=[100, 2200], ro_lims=[0.0, 1.0], datetime_unique=False, method='trf', loss='soft_l1', f_scale=0.1, max_nfev=None, xtol=1e-09*)

Process the raw transient data and perform the Kolber et al. 1998 saturation model.

Parameters

- **pfd** (*np.array, dtype=float, shape=[n,]*) – The photon flux density of the instrument in μmol photons m^2 s^{-1}.
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The fluorescence yield of the instrument.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The measurement number.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date & time of each measurement.
- **blank** (*np.array, dype=float, shape=[n,]*) – The blank value, must be the same length as flevel.
- **sat_len** (*int, default=100*) – The number of flashlets in saturation sequence.
- **skip** (*int, default=0*) – the number of flashlets to skip at start.
- **ro** (*float, default=0.3*) – The fixed value of the connectivity coefficient. Not required if fixed_ro is False.
- **no_ro** (*bool, default=False*) – If True, this processes the raw transient data and performs the no connectivity saturation model.
- **calc_ro** (*bool, default=True*) – If True, this processes the raw transient data and performs the fit with the connectivity saturation model.

- **fixed_ro** (*bool, default=False*) – If True, this sets a user defined fixed value for ro (the connectivity factor) when fitting the saturation model.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.
- **sig_lims** (*[int, int], default=[100, 2200]*) – The lower and upper limit bounds for fitting sigmaPSII.
- **ro_lims** (*[float, float], default=[0.0, 0.1]*) – The lower and upper limit bounds for fitting the connectivity coefficient. Not required if no_ro and fixed_ro are False.
- **datatime_unique** (*bool, default=False*) – If True, will find the unique datetime values for each fit.
- **method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **f_scale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- **res** (*pandas.DataFrame*) – The results of the fitting routine with columns as below:
- **fo** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level.
- **fm** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level.
- **sigma** (*np.array, dtype=float, shape=[n,]*) – The effective absorption cross-section of PSII in ^{2}.
- **fvfm** (*np.array, dtype=float, shape=[n,]*) – The maximum photochemical efficiency.
- **ro** (*np.array, dtype=float, shape=[n,]*) – The connectivity coefficient, ρ, only returned if no_ro and fixed_ro are False.
- **bias** (*np.array, dtype=float, shape=[n,]*) – The bias of fit in %.
- **rmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit.
- **nrmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit normalised to the mean of the fluorescence level.
- **fo_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of F_o in %.
- **fm_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of F_m in %.
- **sigma_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of σ_{PSII}.

- **ro_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of ρ, only returned if no_ro and fixed_ro are False.
- **nfl** (*np.array, dtype=int, shape=[n,]*) – The number of flashlets used for fitting.
- **niters** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the fitting routine.
- **flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -3 : Unable to calculate parameter errors -2 : $F_o > F_m$ -1 : improper input parameters status returned from MINPACK. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time associated with the measurement.

Example

```
>>> sat = ppu.calculate_saturation(pfd, flevel, seq, datetime, blank=0, sat_
↪len=100, skip=0, ro=0.3, no_ro=False, fixed_ro=True, sig_lims =[100,2200])
```

CHAPTER 4

Relaxation

phyto_photo_utils._relaxation.**fit_relaxation**(*flevel, seq_time, seq, datetime, blank=0, sat_len=100, rel_len=60, sat_flashlets=None, single_decay=False, bounds=True, single_lims=[100, 50000], tau1_lims=[100, 800], tau2_lims=[800, 2000], tau3_lims=[2000, 50000], method='trf', loss='soft_l1', f_scale=0.1, max_nfev=None, xtol=1e-09*)

Process the raw transient data and perform the Kolber et al. 1998 relaxation model.

Parameters

- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The sequence time of the flashlets in μs.
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The fluorescence yield of the instrument.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The measurement number.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date & time of each measurement in the numpy datetime64 format.
- **blank** (*np.array, dtype=float, shape=[n,]*) – The blank value, must be the same length as flevel.
- **sat_len** (*int, default=100*) – The number of flashlets in the saturation sequence.
- **rel_len** (*int, default=60*) – The number of flashlets in the relaxation sequence.
- **sat_flashlets** (*int, default=0*) – The number of saturation flashlets to include at the start.
- **single_decay** (*bool, default=False*) – If True, will fit a single decay relaxation.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.

- **single_lims** (*[int, int], default=[100, 50000]*) – The lower and upper limit bounds for fitting τ, only required if single_decay is True.
- **tau1_lims** (*[int, int], default=[100, 800]*) – The lower and upper limit bounds for fitting $\tau 1$.
- **tau2_lims** (*[int, int], default=[800, 2000]*) – The lower and upper limit bounds for fitting $\tau 2$.
- **tau3_lims** (*[int, int], default=[2000, 50000]*) – The lower and upper limit bounds for fitting $\tau 3$.
- **fit_method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss_method** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **fscale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- **res** (*pandas.DataFrame*) – The results of the fitting routine with columns as below:
- **fo_r** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level of relaxation phase.
- **fm_r** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level of relaxation phase
- **tau** (*np.array, dtype=float, shape=[n,]*) – The rate of QA^- reoxidation in μs, only returned if single_decay is True.
- **alpha1** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_1, only returned if single_decay is False.
- **tau1** (*np.array, dtype=float, shape=[n,]*) – The rate of QA^- reoxidation in μs, only returned if single_decay is False.
- **alpha2** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_2.
- **tau2** (*np.array, dtype=float, shape=[n,]*) – The rate of QB^- reoxidation in μs, only returned if single_decay is False.
- **alpha3** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_3, only returned if single_decay is False.
- **tau3** (*np.array, dtype=float, shape=[n,]*) – The rate of PQ reoxidation in μs, only returned if single_decay is False.
- **bias** (*np.array, dtype=float, shape=[n,]*) – The bias of fit in %.

- **rmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit.
- **nrmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit normalised to the mean of the fluorescence level.
- **fo_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of Fo_relax in %.
- **fm_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of Fm_relax in %.
- **tau_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ, only returned if single_decay is True.
- **alpha1_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_1, only returned if single_decay is False.
- **tau1_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_1, only returned if single_decay is False.
- **alpha2_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_2, only returned if single_decay is False.
- **tau2_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_2, only returned if single_decay is False.
- **alpha3_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_3, only returned if single_decay is False.
- **tau3_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_3, only returned if single_decay is False.
- **nfl** (*np.array, dtype=int, shape=[n,]*) – The number of flashlets used for fitting.
- **niters** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the fitting routine.
- **flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -3 : Unable to calculate parameter errors -2 : F_o Relax > F_m Relax -1 : improper input parameters status returned from MINPACK. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time associated with the measurement.

Example

```
>>> rel = ppu.calculate_relaxation(flevel, seq_time, seq, datetime, blank=0, sat_
↪len=100, rel_len=40, single_decay=True, bounds=True, tau_lims=[100, 50000])
```

CHAPTER 5

General Tools

phyto_photo_utils._tools.**calculate_blank_FIRe**(*file_*)

Calculates the blank by averaging the fluorescence yield for the saturation phase.

Parameters **file** (*str*) – The path directory to the raw blank file.

Returns **res** – The blank results: blank, datetime

Return type pandas.DataFrame

Example

```
>>> ppu.calculate_blank_FIRe(file_)
```

phyto_photo_utils._tools.**calculate_blank_FastOcean**(*file_*, *seq_len=100*, *delimiter=','*)

Calculates the blank by averaging the fluorescence yield for the saturation phase.

Parameters

- **file** (*str*) – The path directory to the raw blank file in csv format.
- **seq_len** (*int, default=100*) – The length of the measurement sequence.
- **delimiter** (*str, default=','*) – Specify the delimiter to be used by Pandas.read_csv for loading the raw files.

Returns **res** – The blank results.

Return type pandas.DataFrame

Example

```
>>> ppu.calculate_blank_FastOcean(file_, seq_len=100)
```

phyto_photo_utils._tools.**correct_fire_instrument_bias**(*df*, *pos=1*, *sat_len=100*)

Corrects for instrumentation bias in the relaxation phase by calculating difference between flashlet 0 the relaxation phase & flashlet[pos]. This bias is then added to the relaxation phase.

Parameters

- **df** (*pandas.DataFrame*) – A dataframe of the raw data, can either be imported from pandas.read_csv or the output from phyto_photo_utils.load
- **pos** (*int, default=1*) – The flashlet number after the start of the phase, either saturation or relaxation, to calculate difference between.
- **sat_len** (*int, default=100*) – The length of saturation measurements.

Returns **df** – A dataframe of FIRe data corrected for the instrument bias.

Return type pandas.DataFrame

Example

```
>>> ppu.correct_fire_bias_correction(df, pos=1, sat_len=100)
```

phyto_photo_utils._tools.**remove_outlier_from_time_average**(*df*, *time=4*, *multiplier=3*)

Remove outliers when averaging transients before performing the fitting routines, used to improve the signal to noise ratio in low biomass systems.

The function sets a time window to average over, using upper and lower limits for outlier detection. The upper and lower limits are determined by mean ± std * [1]. The multiplier [1] can be adjusted by the user.

Parameters

- **df** (*pandas.DataFrame*) – A dataframe of the raw data, can either be imported from pandas.read_csv or the output from phyto_photo_utils.load
- **time** (*int, default=4*) – The time window to average over, e.g. 4 = 4 minute averages
- **multiplier** (*int, default=3*) – The multiplier to apply to the standard deviation for determining the upper and lower limits.

Returns **df** – A dataframe of the time averaged data with outliers excluded.

Return type pandas.DataFrame

Example

```
>>> ppu.remove_outlier_from_time_average(df, time=2, multiplier=3)
```

CHAPTER 6

Fluorescence Light Curves

phyto_photo_utils._etr.**calculate_amplitude_etr**(*fo, fm, sigma, par, alpha_phase=True, light_independent=True, dark_sigma=False, etrmax_fitting=True, serodio_sigma=False, light_step_size=None, last_steps_average=False, outlier_multiplier=3, return_data=False, bounds=True, alpha_lims=[0, 4], etrmax_lims=[0, 2000], method='trf', loss='soft_l1', f_scale=0.1, max_nfev=None, xtol=1e-09*)

Convert the processed transient data into an electron transport rate and perform a fit using the Webb Model.

Parameters

- **fo** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level.
- **fm** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level.
- **sigma** (*np.array, dtype=float, shape=[n,]*) – The effective absorption cross-section of PSII in ^{2}.
- **par** (*np.array, dtype=float, shape=[n,]*) – The actinic light levels in μE m^2 s^{-1}.
- **alpha_phase** (*bool, default=True*) – If True, will fit the data without photoinhibition. If False, will fit the data with the photoinhibition paramater β.
- **light_independent** (*bool, default=True*) – If True, will use the method outlined in Silsbe & Kromkamp 2012.
- **dark_sigma** (*bool*) – If True, will use mean of σ_{PSII} under 0 actinic light for calculation. If False, will use σ_{PSII} and σ_{PSII}' for calculation.
- **etrmax_fitting** (*bool*) – If True, will fit α^{ETR} and ETR_{max} and manually calculate E:sub:'k'. If False, will fit α^{ETR} and E:sub:'k' and manually calculate ETR_{max}.

- **serodio_sigma** (*bool*) – If True, will apply a Serodio correction for samples that have dark relaxation.
- **light_step_size** (*int*) – The number of measurements for initial light step.
- **last_steps_average** (*bool, default=False,*) – If True, means will be created from the last 3 measurements per light step. Else, mean will be created from entire light step excluding outliers.
- **outlier_multiplier** (*int, default=3*) – The multiplier to apply to the standard deviation for determining the upper and lower limits.
- **return_data** (*bool, default=False*) – If True, will return the final data used for the fit.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.
- **alpha_lims** (*[int, int], default=[0,4]*) – The lower and upper limit bounds for fitting α^{ETR}.
- **etrmax_lims** (*[int, int], default=[0,2000]*) – The lower and upper limit bounds for fitting ETR_{max}.
- **fit_method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss_method** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **fscale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- *Results are returned as pd.Series with the following parameters.*
- **etr_max** (*float*) – The maximum electron transport rate.
- **alpha** (*float*) – The light limited slope of electron transport.
- **ek** (*float*) – The photoacclimation of ETR.
- **alpha_bias** (*float*) – The bias of the alpha fit. If alpha_phase is False, value is not returned.
- **alpha_rmse** (*float*) – The root mean squared error of the alpha fit. If alpha_phase is False, value is not returned.
- **alpha_nrmse** (*float*) – The normalised root mean squared error of the alpha fit. If alpha_phase is False, value is not returned.
- **beta_bias** (*float*) – The bias of the alpha fit. If alpha_phase is True, value is not returned.

- **beta_rmse** (*float*) – The root mean squared error of the alpha fit. If alpha_phase is True, value is not returned.
- **beta_nrmse** (*float*) – The normalised root mean squared error of the alpha fit. If alpha_phase is True, value is not returned.
- **etrmax_err** (*float*) – The fit error of ETR^{max}. If etrmax_fitting is False, value returned is NaN.
- **alpha_err** (*float*) – The fit error of α_{ETR}.
- **ek_err** (*float*) – The fit error of E_k. If etrmax_fitting is True, value returned is NaN.
- **alpha_nfev** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the alpha phase fitting routine.
- **alpha_flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -1 : the ETR data is empty. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **alpha_success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **beta_nfev** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the beta phase fitting routine. If alpha_phase is True, value returned is NaN.
- **beta_flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. If alpha_phase is True, value returned is NaN. -1 : the ETR data is empty. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **beta_success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE). If alpha_phase is True, value returned is NaN.
- **data** (*[np.array, np.array]*) – Optional, the final data used for the fitting procedure.

Example

```
>>> res = ppu.calculate_etr(fo, fm, sigma, par, return_data=False)
```

CHAPTER 7

Spectral Correction

phyto_photo_utils._spectral_correction.**calculate_chl_specific_absorption**(*aptot*, *blank*, *ap_lambda*, *depig=None*, *chl=None*, *vol=None*, *betac1=None*, *betac2=None*, *diam=None*, *bricaud_slope=False*, *phycobilin=False*, *norm_750=False*)

Process the raw absorbance data to produce chlorophyll specific phytoplankton absorption.

Parameters

- **aptot** (*np.array, dtype=float, shape=[n,]*) – The raw absorbance data.
- **blank** (*np.array, dtype=float, shape=[n,]*) – The blank absorbance data.
- **ap_lambda** (*np.array, dtype=float, shape=[n,]*) – The wavelengths corresponding to the measurements.
- **depig** (*np.array, dtype=float, shape=[n,]*) – The raw depigmented absorbance data.
- **chl** (*float*) – The chlorophyll concentration associated with the measurement.
- **vol** (*int*) – The volume of water filtered in mL.

- **betac1** (*int*) – The pathlength amplification coefficient 1 (see Stramski et al. 2015). For transmittance mode, 0.679, for transmittance-reflectance mode, 0.719, and for integrating sphere mode, 0.323.
- **betac2** (*int*) – The pathlength amplification coefficient 2 (see Stramski et al. 2015). For transmittance mode, 1.2804, for transmittance-reflectance mode, 1.2287, and for integrating sphere mode, 1.0867.
- **diam** (*float*) – The diameter of filtrate in mm.
- **bricaud_slope** (*bool, default=True*) – If True, will theoretically calculate detrital slope (see Bricaud & Stramski 1990). If False, will subtract depigmented absorption from total absorption.
- **phycobilin** (*bool, default=False*) – If True, will account for high absorption in the green wavelengths (580 - 600 nm) by phycobilin proteins when performing the bricaud_slope detrital correction.
- **norm_750** (*bool, default=False*) – If True, will normalise the data to the value at 750 nm.

Returns **aphy** – The chlorophyll specific phytoplankton absorption data.

Return type np.array, dtype=float, shape=[n,]

Example

```
>>> aphy = ppu.calculate_chl_specific_absorption(pa_data, blank, wavelength,␣
↪chl=0.19, vol=2000, betac1=0.323, betac2=1.0867, diam=15, bricaud_slope=True,␣
↪phycobilin=True, norm750=False)
```

phyto_photo_utils._spectral_correction.**calculate_instrument_led_correction**(*aphy*, *ap_lambda*, *method=None*, *chl=None*, *e_background=None*, *e_insitu=None*, *e_actinic=None*, *depth=None*, *e_led=None*, *wl=None*, *constants=None*)

Calculate the spectral correction factor TO DO: Create method to convert all arrays to same length and resolution TO DO: Add in functionality to calculate mixed excitation wavelength spectra for FastOcean when using more than wavelength

Parameters

- **aphy** (*np.array, dtype=float, shape=[n,]*) – The wavelength specific phytoplankton absorption coefficients.
- **ap_lambda** (*np.array, dtype=int, shape=[n,]*) – The wavelengths associated with the aphy and aphy_star.
- **method** (*'sigma', 'actinic'*) – Choose spectral correction method to either correct SigmaPSII or correct the background actinic light.

- **e_background** (*'insitu', 'actinic'*) – For sigma spectral correction factor, select either insitu light (e.g. for underway or insitu measurements) or actinic light (e.g. for fluorescence light curves) as the background light source
- **e_insitu** (*np.array, dtype=int, shape=[n,]*) – The in situ irradiance field, if None is passed then will theoretically calculate in situ light field.
- **chl** (*dtype=float*) – Chlorophyll concentration for estimation of Kbio for theoretical in situ light field. If None is passed then chl is set to 1 mg/m3.
- **e_actinic** (*'fastact'*) – Actinic light spectrum e.g. Spectra of the Actinic lights within the FastAct illuminating during Fluorescence Light Curves etc. Must be defined for 'actinic' method.
- **depth** (*float, default=None*) – The depth of the measurement. Must be set if theoretically calculating e_insitu.
- **e_led** (*'fire','fasttracka_ii', 'fastocean'*) – The excitation spectra of the instrument.
- **wl** (*'450nm', '530nm', 624nm', None*) – For FastOcean only. Select the excitation wavelength. Future PPU versions will provide option to mix LEDs.
- **constants** (*file, default=None*) – The spectra of the light source used for the measurement. If None is provided, will read in default values stored within PPU.

Returns **scf** – The spectral correction factor to correct SigmaPSII or actinic background light depending on method.

Return type float

Example

```
>>> ppu.calculate_instrument_led_correction(aphy, wavelength, e_led='fire')
```

CHAPTER 8

Plotting

phyto_photo_utils._plot.**plot_fluorescence_light_curve**(*par*, *etr*, *etrmax=None*, *alpha=None*, *rmse=None*, *sigma=None*, *phi=False*)

Parameters

- **par** (*np.array, dtype=float*) – The actinic light data from the fluorescence light curve.
- **etr** (*np.array, dtype=float*) – The electron transport rate data.
- **etrmax** (*float, default=None*) – The maximum electron transport rate.
- **alpha** (*float, default=None*) – The light limited slope of electron transport.
- **rmse** (*float, default=None*) – The RMSE value of the fit.
- **sigma** (*float, default=None*) – The effective absorption-cross section.
- **phi** (*bool, default=False*) – If True, etr data is phi and the modified Webb et al. (1974) fit is used.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> ppu.plot_fluorescence_light_curve(par, etr, etrmax=etr_max, alpha=alpha,␣
↪rsq=rsq, sigma=sigma, phi=True)
```

phyto_photo_utils._plot.**plot_relaxation_data**(*fyield*, *seq_time*, *fo_relax=None*, *fm_relax=None*, *tau=None*, *alpha=None*, *rmse=None*)

Parameters

- **fyield** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence yield data.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The time of the flashlet measurements.
- **fo_relax** (*float, default=None*) – The minimum fluorescence value in the relaxation phase.
- **fm_relax** (*float, default=None*) – The maximum fluorescence value in the relaxation phase.
- **tau** (*float, default=None*) – The rate of reoxidation in μs.
- **alpha** (*float, default=None*) – The ratio of reoxidisation components.
- **rmse** (*float, default=None*) – The RMSE value of the fit.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> ppu.plot_relaxation_data(fyield, seq_time, fo_relax=fo_r, fm_relax=fm_r,␣
↪tau=(tau1, tau2, tau3), alpha=(alpha1, alpha2, alpha3), rsq=rsq)
```

phyto_photo_utils._plot.**plot_saturation_data**(*fyield*, *pfd*, *fo=None*, *fm=None*, *sigma=None*, *ro=None*, *rmse=None*)

Parameters

- **fyield** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence yield data.
- **pfd** (*np.array, dtype=float, shape=[n,]*) – The photon flux density.
- **fo** (*float, default=None*) – The minimum fluorescence value.
- **fm** (*float, default=None*) – The maximum fluorescence value.
- **sigma** (*float, default=None*) – The effective absorption cross-section value in ^{2}.
- **ro** (*float, default=None*) – The connectivity coefficient.
- **rmse** (*float, default=None*) – The RMSE value of the fit.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> plot_saturation_data(fyield, pfd, fo=fo, fm=fm, sigma=sigma, ro=None,␣
↪rmse=rmse)
```

CHAPTER 9

API Documentation

9.1 Loading

phyto_photo_utils._load.**load_FASTTrackaI_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=120*, *irrad=None*)

Process the raw data file and convert to a csv with standard formatting.

Parameters

- **file** (*str*) – The path directory to the .000 data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*dir, default=None*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=120*) – The number of flashlets in the protocol.
- **irrad** (*int, default=None*) – The light/dark chamber photons per count from the calibration file.

Returns

- **df** (*pandas.DataFrame, shape=[n,8]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.

- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.
- **channel** (*np.array, dtype=str, shape=[n,]*) – The chamber used for measurements, A = light chamber, B = dark chamber.
- **gain** (*np.array, dtype=int, shape=[n,]*) – The gain settings of the instrument.

Example

```
>>> fname = './data/raw/instrument/fasttrackai/FASTTrackaI_example.csv'
>>> output = './data/raw/ppu/fasttrackai/'
>>> df = ppu.load_FASTTrackaI_files(fname, append=False, save_files=True, res_
↪path=output, seq_len=120, irrad=545.62e10)
```

phyto_photo_utils._load.**load_FIRe_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=160*, *gain_value=None*, *flen=1e-06*, *irrad=None*, *continuous=False*, *light_step=False*, *single_turnover=True*)

Process the raw data file (.000 format) and convert to a csv with standard formatting.

Parameters

- **file** (*str*) – The path directory to the data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*str, default=None*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int , default=160*) – The number of flashlets in the protocol. Only required if continuous = True.
- **gain** (*int, default=None*) – The gain value set when performing measurements. Only required if continous = True. Discrete Gain values are recorded in raw data files.
- **flen** (*float, default=1e-6*) – The flashlet length in seconds.
- **irrad** (*int, default=None*) – The LED output in μE m^2 s^{-1}. Only required if continuous = True.
- **continuous** (*bool, default=False*) – If True, will load files from the continuous format. If False, will load the discrete file format.
- **light_step** (*bool, default=False*) – If True, will load files from a discrete format FLC file. If False, will load the discrete file format with no light steps.
- **single_turnover** (*bool, default=True*) – If True, will load the saturation and relaxation from the single turnover measurement. If False, will load the multiple turnover measurement.

Returns

- **df** (*pandas.DataFrame, shape=[n,6]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to seq_len

- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.

Example

```
>>> fname = './data/raw/instrument/fire/FIRe_example.000'
>>> output = './data/raw/ppu/fire/'
>>> df = ppu.load_FIRe_files(fname, append=False, save_files=True, res_
↪path=output, seq_len=160, flen=1e-6, irrad=47248)
```

phyto_photo_utils._load.**load_FastOcean_files**(*file_*, *append=False*, *save_files=False*, *led_separate=False*, *res_path=None*, *seq_len=125*, *seq_reps=None*, *flen=1e-06*, *delimiter=','*, *FastAct1=True*, *Single_Acq=False*)

Process the raw data file and convert to a csv with standard formatting.

Parameters

- **file** (*dir*) – The path directory to the .csv data file from the FastOcean with either the FastAct11 or FastAct12 laboratory system.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together. Not applicable if Single_Acq = True.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **led_separate** (*bool, default=False*) – If True, the protocols will be separated dependent upon the LED sequence.
- **res_path** (*dir*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=125*) – The number of flashlets in the protocol.
- **seq_reps** (*int, default=None*) – The number of sequences per acquisition or in FastAct2 multiple acquisition files the number of light steps.
- **flen** (*float, default=1e-6*) – The flashlet length in seconds.
- **delimiter** (*str, default=','*) – Specify the delimiter to be used by Pandas.read_csv for loading the raw files.
- **FastAct1** (*bool, default=True*) – If True, will load data from FastAct1 laboratory system format. If False, will load data from FastAct2 laboratory system format.
- **Single_Acq** (*bool, default=False*) – If True, will load a single acquisition data from either the FastAct1 or FastAct2 laboratory system, dependent upon whether FastAct1 is True of False.

Returns

- **df** (*pandas.DataFrame, shape=[n,7]*) – A dataframe of the raw fluorescence data with columns as below:

- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.
- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.
- **led_sequence** (*np.array, dtype=int, shape=[n,]*) – The LED combination using during the measurement, see example below.

Example

```
>>> fname = './data/raw/instrument/fire/FastOcean_example.csv'
>>> output = './data/raw/ppu/fastocean/'
>>> df = ppu.load_FastOcean_files(fname, append=False, save_files=True, led_
↪separate=False, res_path=output, seq_len=125, flen=1e-6)
>>> led_sequence == 1, LED 450 nm
>>> led_sequence == 2, LED 450 nm + LED 530 nm
>>> led_sequence == 3, LED 450 nm + LED 624 nm
>>> led_sequence == 4, LED 450 nm + LED 530 nm + LED 624 nm
>>> led_sequence == 5, LED 530 nm + LED 624 nm
>>> led_sequence == 6, LED 530 nm
>>> led_sequence == 7, LED 624 nm
```

phyto_photo_utils._load.**load_LIFT_FRR_files**(*file_*, *append=False*, *save_files=False*, *res_path=None*, *seq_len=228*)

Process the raw data file from the Soliense LIFT FRR and convert to a csv with standard formatting.

Parameters

- **file** (*dir*) – The path directory to the .000 data file from benchtop SAtlantic FIRe.
- **append** (*bool, default=False*) – If True, multiple files will be concatenated together.
- **save_files** (*bool, default=False*) – If True, files will be saved as .csv.
- **res_path** (*dir*) – The path directory where to save files, only required if save_files = True.
- **seq_len** (*int, default=228*) – The number of flashlets in the protocol.

Returns

- **df** (*pandas.DataFrame, shape=[n,7]*) – A dataframe of the raw fluorescence data with columns as below:
- **flashlet_number** (*np.array, dtype=int, shape=[n,]*) – A sequential number from 1 to `seq_len`
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence data.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time of measurement.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The sample measurement number.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The measurement sequence time in μs.

- **pfd** (*np.array, dype=float, shape=[n,]*) – The photon flux density in μmol photons m^2 s^{-1}.

9.2 General Tools

phyto_photo_utils._tools.**calculate_blank_FIRe**(*file_*)

Calculates the blank by averaging the fluorescence yield for the saturation phase.

Parameters **file** (*str*) – The path directory to the raw blank file.

Returns **res** – The blank results: blank, datetime

Return type pandas.DataFrame

Example

```
>>> ppu.calculate_blank_FIRe(file_)
```

phyto_photo_utils._tools.**calculate_blank_FastOcean**(*file_*, *seq_len=100*, *delimiter=','*)

Calculates the blank by averaging the fluorescence yield for the saturation phase.

Parameters

- **file** (*str*) – The path directory to the raw blank file in csv format.
- **seq_len** (*int, default=100*) – The length of the measurement sequence.
- **delimiter** (*str, default=','*) – Specify the delimiter to be used by Pandas.read_csv for loading the raw files.

Returns **res** – The blank results.

Return type pandas.DataFrame

Example

```
>>> ppu.calculate_blank_FastOcean(file_, seq_len=100)
```

phyto_photo_utils._tools.**correct_fire_instrument_bias**(*df*, *pos=1*, *sat_len=100*)

Corrects for instrumentation bias in the relaxation phase by calculating difference between flashlet 0 the relaxation phase & flashlet[pos]. This bias is then added to the relaxation phase.

Parameters

- **df** (*pandas.DataFrame*) – A dataframe of the raw data, can either be imported from pandas.read_csv or the output from phyto_photo_utils.load
- **pos** (*int, default=1*) – The flashlet number after the start of the phase, either saturation or relaxation, to calculate difference between.
- **sat_len** (*int, default=100*) – The length of saturation measurements.

Returns **df** – A dataframe of FIRe data corrected for the instrument bias.

Return type pandas.DataFrame

Example

```
>>> ppu.correct_fire_bias_correction(df, pos=1, sat_len=100)
```

phyto_photo_utils._tools.**remove_outlier_from_time_average**(*df*, *time=4*, *multiplier=3*)

Remove outliers when averaging transients before performing the fitting routines, used to improve the signal to noise ratio in low biomass systems.

The function sets a time window to average over, using upper and lower limits for outlier detection. The upper and lower limits are determined by mean ± std * [1]. The multiplier [1] can be adjusted by the user.

Parameters

- **df** (*pandas.DataFrame*) – A dataframe of the raw data, can either be imported from pandas.read_csv or the output from phyto_photo_utils.load
- **time** (*int, default=4*) – The time window to average over, e.g. 4 = 4 minute averages
- **multiplier** (*int, default=3*) – The multiplier to apply to the standard deviation for determining the upper and lower limits.

Returns **df** – A dataframe of the time averaged data with outliers excluded.

Return type pandas.DataFrame

Example

```
>>> ppu.remove_outlier_from_time_average(df, time=2, multiplier=3)
```

9.3 Saturation

phyto_photo_utils._saturation.**fit_saturation**(*pfd*, *flevel*, *seq*, *datetime*, *blank=0*, *sat_len=100*, *skip=0*, *ro=0.3*, *no_ro=False*, *calc_ro=True*, *fixed_ro=False*, *bounds=True*, *sig_lims=[100, 2200]*, *ro_lims=[0.0, 1.0]*, *datetime_unique=False*, *method='trf'*, *loss='soft_l1'*, *f_scale=0.1*, *max_nfev=None*, *xtol=1e-09*)

Process the raw transient data and perform the Kolber et al. 1998 saturation model.

Parameters

- **pfd** (*np.array, dtype=float, shape=[n,]*) – The photon flux density of the instrument in μmol photons m^2 s^{-1}.
- **flevel** (*np.array, dtype=float, shape=[n,]*) – The fluorescence yield of the instrument.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The measurement number.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date & time of each measurement.
- **blank** (*np.array, dype=float, shape=[n,]*) – The blank value, must be the same length as flevel.

- **sat_len** (*int, default=100*) – The number of flashlets in saturation sequence.
- **skip** (*int, default=0*) – the number of flashlets to skip at start.
- **ro** (*float, default=0.3*) – The fixed value of the connectivity coefficient. Not required if fixed_ro is False.
- **no_ro** (*bool, default=False*) – If True, this processes the raw transient data and performs the no connectivity saturation model.
- **calc_ro** (*bool, default=True*) – If True, this processes the raw transient data and performs the fit with the connectivity saturation model.
- **fixed_ro** (*bool, default=False*) – If True, this sets a user defined fixed value for ro (the connectivity factor) when fitting the saturation model.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.
- **sig_lims** (*[int, int], default=[100, 2200]*) – The lower and upper limit bounds for fitting sigmaPSII.
- **ro_lims** (*[float, float], default=[0.0, 0.1]*) – The lower and upper limit bounds for fitting the connectivity coefficient. Not required if no_ro and fixed_ro are False.
- **datatime_unique** (*bool, default=False*) – If True, will find the unique datetime values for each fit.
- **method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **f_scale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- **res** (*pandas.DataFrame*) – The results of the fitting routine with columns as below:
- **fo** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level.
- **fm** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level.
- **sigma** (*np.array, dtype=float, shape=[n,]*) – The effective absorption cross-section of PSII in ^{2}.
- **fvfm** (*np.array, dtype=float, shape=[n,]*) – The maximum photochemical efficiency.

- **ro** (*np.array, dtype=float, shape=[n,]*) – The connectivity coefficient, ρ, only returned if no_ro and fixed_ro are False.
- **bias** (*np.array, dtype=float, shape=[n,]*) – The bias of fit in %.
- **rmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit.
- **nrmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit normalised to the mean of the fluorescence level.
- **fo_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of F_o in %.
- **fm_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of F_m in %.
- **sigma_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of σ_{PSII}.
- **ro_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of ρ, only returned if no_ro and fixed_ro are False.
- **nfl** (*np.array, dtype=int, shape=[n,]*) – The number of flashlets used for fitting.
- **niters** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the fitting routine.
- **flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -3 : Unable to calculate parameter errors -2 : $F_o > F_m$ -1 : improper input parameters status returned from MINPACK. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time associated with the measurement.

Example

```
>>> sat = ppu.calculate_saturation(pfd, flevel, seq, datetime, blank=0, sat_
↪len=100, skip=0, ro=0.3, no_ro=False, fixed_ro=True, sig_lims =[100,2200])
```

9.4 Relaxation

phyto_photo_utils._relaxation.**fit_relaxation**(*flevel, seq_time, seq, datetime, blank=0, sat_len=100, rel_len=60, sat_flashlets=None, single_decay=False, bounds=True, single_lims=[100, 50000], tau1_lims=[100, 800], tau2_lims=[800, 2000], tau3_lims=[2000, 50000], method='trf', loss='soft_l1', f_scale=0.1, max_nfev=None, xtol=1e-09*)

Process the raw transient data and perform the Kolber et al. 1998 relaxation model.

Parameters

- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The sequence time of the flashlets in μs.

- **flevel** (*np.array, dtype=float, shape=[n,]*) – The fluorescence yield of the instrument.
- **seq** (*np.array, dtype=int, shape=[n,]*) – The measurement number.
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date & time of each measurement in the numpy datetime64 format.
- **blank** (*np.array, dtype=float, shape=[n,]*) – The blank value, must be the same length as flevel.
- **sat_len** (*int, default=100*) – The number of flashlets in the saturation sequence.
- **rel_len** (*int, default=60*) – The number of flashlets in the relaxation sequence.
- **sat_flashlets** (*int, default=0*) – The number of saturation flashlets to include at the start.
- **single_decay** (*bool, default=False*) – If True, will fit a single decay relaxation.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.
- **single_lims** (*[int, int], default=[100, 50000]*) – The lower and upper limit bounds for fitting τ, only required if single_decay is True.
- **tau1_lims** (*[int, int], default=[100, 800]*) – The lower and upper limit bounds for fitting $\tau 1$.
- **tau2_lims** (*[int, int], default=[800, 2000]*) – The lower and upper limit bounds for fitting $\tau 2$.
- **tau3_lims** (*[int, int], default=[2000, 50000]*) – The lower and upper limit bounds for fitting $\tau 3$.
- **fit_method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss_method** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **fscale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- **res** (*pandas.DataFrame*) – The results of the fitting routine with columns as below:
- **fo_r** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level of relaxation phase.

- **fm_r** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level of relaxation phase
- **tau** (*np.array, dtype=float, shape=[n,]*) – The rate of QA^- reoxidation in μs, only returned if single_decay is True.
- **alpha1** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_1, only returned if single_decay is False.
- **tau1** (*np.array, dtype=float, shape=[n,]*) – The rate of QA^- reoxidation in μs, only returned if single_decay is False.
- **alpha2** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_2.
- **tau2** (*np.array, dtype=float, shape=[n,]*) – The rate of QB^- reoxidation in μs, only returned if single_decay is False.
- **alpha3** (*np.array, dtype=float, shape=[n,]*) – The decay coefficient of τ_3, only returned if single_decay is False.
- **tau3** (*np.array, dtype=float, shape=[n,]*) – The rate of PQ reoxidation in μs, only returned if single_decay is False.
- **bias** (*np.array, dtype=float, shape=[n,]*) – The bias of fit in %.
- **rmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit.
- **nrmse** (*np.array, dtype=float, shape=[n,]*) – The root mean squared error of the fit normalised to the mean of the fluorescence level.
- **fo_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of Fo_relax in %.
- **fm_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of Fm_relax in %.
- **tau_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ, only returned if single_decay is True.
- **alpha1_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_1, only returned if single_decay is False.
- **tau1_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_1, only returned if single_decay is False.
- **alpha2_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_2, only returned if single_decay is False.
- **tau2_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_2, only returned if single_decay is False.
- **alpha3_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of α_3, only returned if single_decay is False.
- **tau3_err** (*np.array, dtype=float, shape=[n,]*) – The fit error of τ_3, only returned if single_decay is False.
- **nfl** (*np.array, dtype=int, shape=[n,]*) – The number of flashlets used for fitting.
- **niters** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the fitting routine.
- **flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -3 : Unable to calculate parameter errors -2 : F_o Relax > F_m Relax -1 : improper input parameters status returned

from MINPACK. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.

- **success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **datetime** (*np.array, dtype=datetime64, shape=[n,]*) – The date and time associated with the measurement.

Example

```
>>> rel = ppu.calculate_relaxation(flevel, seq_time, seq, datetime, blank=0, sat_
↪len=100, rel_len=40, single_decay=True, bounds=True, tau_lims=[100, 50000])
```

9.5 Spectral Correction

phyto_photo_utils._spectral_correction.**calculate_chl_specific_absorption**(*aptot*, *blank*, *ap_lambda*, *depig=None*, *chl=None*, *vol=None*, *betac1=None*, *betac2=None*, *diam=None*, *bricaud_slope=False*, *phycobilin=False*, *norm_750=False*)

Process the raw absorbance data to produce chlorophyll specific phytoplankton absorption.

Parameters

- **aptot** (*np.array, dtype=float, shape=[n,]*) – The raw absorbance data.
- **blank** (*np.array, dtype=float, shape=[n,]*) – The blank absorbance data.
- **ap_lambda** (*np.array, dtype=float, shape=[n,]*) – The wavelengths corresponding to the measurements.
- **depig** (*np.array, dtype=float, shape=[n,]*) – The raw depigmented absorbance data.
- **chl** (*float*) – The chlorophyll concentration associated with the measurement.
- **vol** (*int*) – The volume of water filtered in mL.
- **betac1** (*int*) – The pathlength amplification coefficient 1 (see Stramski et al. 2015). For transmittance mode, 0.679, for transmittance-reflectance mode, 0.719, and for integrating sphere mode, 0.323.

- **betac2** (*int*) – The pathlength amplification coefficient 2 (see Stramski et al. 2015). For transmittance mode, 1.2804, for transmittance-reflectance mode, 1.2287, and for integrating sphere mode, 1.0867.
- **diam** (*float*) – The diameter of filtrate in mm.
- **bricaud_slope** (*bool, default=True*) – If True, will theoretically calculate detrital slope (see Bricaud & Stramski 1990). If False, will subtract depigmented absorption from total absorption.
- **phycobilin** (*bool, default=False*) – If True, will account for high absorption in the green wavelengths (580 - 600 nm) by phycobilin proteins when performing the bricaud_slope detrital correction.
- **norm_750** (*bool, default=False*) – If True, will normalise the data to the value at 750 nm.

Returns **aphy** – The chlorophyll specific phytoplankton absorption data.

Return type np.array, dtype=float, shape=[n,]

Example

```
>>> aphy = ppu.calculate_chl_specific_absorption(pa_data, blank, wavelength,␣
↪chl=0.19, vol=2000, betac1=0.323, betac2=1.0867, diam=15, bricaud_slope=True,␣
↪phycobilin=True, norm750=False)
```

phyto_photo_utils._spectral_correction.**calculate_instrument_led_correction**(*aphy*, *ap_lambda*, *method=None*, *chl=None*, *e_background=None*, *e_insitu=None*, *e_actinic=None*, *depth=None*, *e_led=None*, *wl=None*, *constants=None*)

Calculate the spectral correction factor TO DO: Create method to convert all arrays to same length and resolution TO DO: Add in functionality to calculate mixed excitation wavelength spectra for FastOcean when using more than wavelength

Parameters

- **aphy** (*np.array, dtype=float, shape=[n,]*) – The wavelength specific phytoplankton absorption coefficients.
- **ap_lambda** (*np.array, dtype=int, shape=[n,]*) – The wavelengths associated with the aphy and aphy_star.
- **method** (*'sigma', 'actinic'*) – Choose spectral correction method to either correct SigmaPSII or correct the background actinic light.
- **e_background** (*'insitu', 'actinic'*) – For sigma spectral correction factor, select either insitu light (e.g. for underway or insitu measurements) or actinic light (e.g. for fluorescence light curves) as the background light source

- **e_insitu** (*np.array, dtype=int, shape=[n,]*) – The in situ irradiance field, if None is passed then will theoretically calculate in situ light field.
- **chl** (*dtype=float*) – Chlorophyll concentration for estimation of Kbio for theoretical in situ light field. If None is passed then chl is set to 1 mg/m3.
- **e_actinic** (*'fastact'*) – Actinic light spectrum e.g. Spectra of the Actinic lights within the FastAct illuminating during Fluorescence Light Curves etc. Must be defined for 'actinic' method.
- **depth** (*float, default=None*) – The depth of the measurement. Must be set if theoretically calculating e_insitu.
- **e_led** (*'fire','fasttracka_ii', 'fastocean'*) – The excitation spectra of the instrument.
- **wl** (*'450nm', '530nm', 624nm', None*) – For FastOcean only. Select the excitation wavelength. Future PPU versions will provide option to mix LEDs.
- **constants** (*file, default=None*) – The spectra of the light source used for the measurement. If None is provided, will read in default values stored within PPU.

Returns **scf** – The spectral correction factor to correct SigmaPSII or actinic background light depending on method.

Return type float

Example

```
>>> ppu.calculate_instrument_led_correction(aphy, wavelength, e_led='fire')
```

9.6 Fluorescence Light Curves

phyto_photo_utils._etr.**calculate_amplitude_etr**(*fo, fm, sigma, par, alpha_phase=True, light_independent=True, dark_sigma=False, etr-max_fitting=True, sero-dio_sigma=False, light_step_size=None, last_steps_average=False, out-lier_multiplier=3, return_data=False, bounds=True, alpha_lims=[0, 4], etr-max_lims=[0, 2000], method='trf', loss='soft_l1', f_scale=0.1, max_nfev=None, xtol=1e-09*)

Convert the processed transient data into an electron transport rate and perform a fit using the Webb Model.

Parameters

- **fo** (*np.array, dtype=float, shape=[n,]*) – The minimum fluorescence level.
- **fm** (*np.array, dtype=float, shape=[n,]*) – The maximum fluorescence level.
- **sigma** (*np.array, dtype=float, shape=[n,]*) – The effective absorption cross-section of PSII in ^{2}.
- **par** (*np.array, dtype=float, shape=[n,]*) – The actinic light levels in μE m^2 s^{-1}.

- **alpha_phase** (*bool, default=True*) – If True, will fit the data without photoinhibition. If False, will fit the data with the photoinhibition paramater β.
- **light_independent** (*bool, default=True*) – If True, will use the method outlined in Silsbe & Kromkamp 2012.
- **dark_sigma** (*bool*) – If True, will use mean of σ_{PSII} under 0 actinic light for calculation. If False, will use σ_{PSII} and σ_{PSII}' for calculation.
- **etrmax_fitting** (*bool*) – If True, will fit α^{ETR} and ETR_{max} and manually calculate E:sub:'k'. If False, will fit α^{ETR} and E:sub:'k' and manually calculate ETR_{max}.
- **serodio_sigma** (*bool*) – If True, will apply a Serodio correction for samples that have dark relaxation.
- **light_step_size** (*int*) – The number of measurements for initial light step.
- **last_steps_average** (*bool, default=False,*) – If True, means will be created from the last 3 measurements per light step. Else, mean will be created from entire light step excluding outliers.
- **outlier_multiplier** (*int, default=3*) – The multiplier to apply to the standard deviation for determining the upper and lower limits.
- **return_data** (*bool, default=False*) – If True, will return the final data used for the fit.
- **bounds** (*bool, default=True*) – If True, will set lower and upper limit bounds for the estimation, not suitable for methods 'lm'.
- **alpha_lims** (*[int, int], default=[0,4]*) – The lower and upper limit bounds for fitting α^{ETR}.
- **etrmax_lims** (*[int, int], default=[0,2000]*) – The lower and upper limit bounds for fitting ETR_{max}.
- **fit_method** (*str, default='trf'*) – The algorithm to perform minimization. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **loss_method** (*str, default='soft_l1'*) – The loss function to be used. Note: Method 'lm' supports only 'linear' loss. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **fscale** (*float, default=0.1*) – The soft margin value between inlier and outlier residuals. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **max_nfev** (*int, default=None*) – The number of iterations to perform fitting routine. If None, the value is chosen automatically. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.
- **xtol** (*float, default=1e-9*) – The tolerance for termination by the change of the independent variables. See `scipy.optimize.least_squares` documentation for more information on non-linear least squares fitting options.

Returns

- *Results are returned as pd.Series with the following parameters.*
- **etr_max** (*float*) – The maximum electron transport rate.
- **alpha** (*float*) – The light limited slope of electron transport.

- **ek** (*float*) – The photoacclimation of ETR.
- **alpha_bias** (*float*) – The bias of the alpha fit. If alpha_phase is False, value is not returned.
- **alpha_rmse** (*float*) – The root mean squared error of the alpha fit. If alpha_phase is False, value is not returned.
- **alpha_nrmse** (*float*) – The normalised root mean squared error of the alpha fit. If alpha_phase is False, value is not returned.
- **beta_bias** (*float*) – The bias of the alpha fit. If alpha_phase is True, value is not returned.
- **beta_rmse** (*float*) – The root mean squared error of the alpha fit. If alpha_phase is True, value is not returned.
- **beta_nrmse** (*float*) – The normalised root mean squared error of the alpha fit. If alpha_phase is True, value is not returned.
- **etrmax_err** (*float*) – The fit error of ETR^{max}. If etrmax_fitting is False, value returned is NaN.
- **alpha_err** (*float*) – The fit error of α_{ETR}.
- **ek_err** (*float*) – The fit error of E_k. If etrmax_fitting is True, value returned is NaN.
- **alpha_nfev** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the alpha phase fitting routine.
- **alpha_flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. -1 : the ETR data is empty. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **alpha_success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE)
- **beta_nfev** (*np.array, dype=int, shape=[n,]*) – The number of functional evaluations done on the beta phase fitting routine. If alpha_phase is True, value returned is NaN.
- **beta_flag** (*np.array, dtype=int, shape=[n,]*) – The code associated with the fitting routine success, positive values = SUCCESS, negative values = FAILURE. If alpha_phase is True, value returned is NaN. -1 : the ETR data is empty. 0 : the maximum number of function evaluations is exceeded. 1 : gtol termination condition is satisfied. 2 : ftol termination condition is satisfied. 3 : xtol termination condition is satisfied. 4 : Both ftol and xtol termination conditions are satisfied.
- **beta_success** (*np.array, dtype=bool, shape=[n,]*) – A boolean array reporting whether fit was successful (TRUE) or if not successful (FALSE). If alpha_phase is True, value returned is NaN.
- **data** (*[np.array, np.array]*) – Optional, the final data used for the fitting procedure.

Example

```
>>> res = ppu.calculate_etr(fo, fm, sigma, par, return_data=False)
```

9.7 Plotting

phyto_photo_utils._plot.**plot_fluorescence_light_curve**(*par*, *etr*, *etrmax=None*, *alpha=None*, *rmse=None*, *sigma=None*, *phi=False*)

Parameters

- **par** (*np.array, dtype=float*) – The actinic light data from the fluorescence light curve.
- **etr** (*np.array, dtype=float*) – The electron transport rate data.
- **etrmax** (*float, default=None*) – The maximum electron transport rate.
- **alpha** (*float, default=None*) – The light limited slope of electron transport.
- **rmse** (*float, default=None*) – The RMSE value of the fit.
- **sigma** (*float, default=None*) – The effective absorption-cross section.
- **phi** (*bool, default=False*) – If True, etr data is phi and the modified Webb et al. (1974) fit is used.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> ppu.plot_fluorescence_light_curve(par, etr, etrmax=etr_max, alpha=alpha,␣
↪rsq=rsq, sigma=sigma, phi=True)
```

phyto_photo_utils._plot.**plot_relaxation_data**(*fyield*, *seq_time*, *fo_relax=None*, *fm_relax=None*, *tau=None*, *alpha=None*, *rmse=None*)

Parameters

- **fyield** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence yield data.
- **seq_time** (*np.array, dtype=float, shape=[n,]*) – The time of the flashlet measurements.
- **fo_relax** (*float, default=None*) – The minimum fluorescence value in the relaxation phase.
- **fm_relax** (*float, default=None*) – The maximum fluorescence value in the relaxation phase.
- **tau** (*float, default=None*) – The rate of reoxidation in μs.
- **alpha** (*float, default=None*) – The ratio of reoxidisation components.
- **rmse** (*float, default=None*) – The RMSE value of the fit.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> ppu.plot_relaxation_data(fyield, seq_time, fo_relax=fo_r, fm_relax=fm_r,␣
↪tau=(tau1, tau2, tau3), alpha=(alpha1, alpha2, alpha3), rsq=rsq)
```

phyto_photo_utils._plot.**plot_saturation_data**(*fyield*, *pfd*, *fo=None*, *fm=None*, *sigma=None*, *ro=None*, *rmse=None*)

Parameters

- **fyield** (*np.array, dtype=float, shape=[n,]*) – The raw fluorescence yield data.
- **pfd** (*np.array, dtype=float, shape=[n,]*) – The photon flux density.
- **fo** (*float, default=None*) – The minimum fluorescence value.
- **fm** (*float, default=None*) – The maximum fluorescence value.
- **sigma** (*float, default=None*) – The effective absorption cross-section value in ^{2}.
- **ro** (*float, default=None*) – The connectivity coefficient.
- **rmse** (*float, default=None*) – The RMSE value of the fit.

Returns **ax** – A matplotlib figure object.

Return type object

Example

```
>>> plot_saturation_data(fyield, pfd, fo=fo, fm=fm, sigma=sigma, ro=None,␣
↪rmse=rmse)
```

CHAPTER 10

Citing and Referencing

10.1 Reference

If you would like to cite or reference Phytoplankton Photophysiology Utilities, please use:

Source: phyto_photo_utils https://gitlab.com/tryankeogh/phytophotoutils retrieved on 30 May 2019.

10.2 Authors and contributors

Thomas Ryan-Keogh: Southern Ocean Carbon and Climate Observatory

Charlotte Robinson: Curtin University

CHAPTER 11

Change Log

v1.4.6(2021-04-12) - Bias metric no longer log transformed. - FIRe loading function includes gain correction and calculations for σ_{PSII} in absolute units.

v1.4.5(2021-01-23) - Bug fix for separating files when loading FastOcean FastAct1 data

v1.4.4(2020-10-21) - Error reporting for ETR calculations cleaned up to reduce redundancy

v1.4.3(2020-10-07) - Bug fix to absorption calculations - Error reporting for ETR calculations

v1.4.2(2020-09-28) - Bug fix for columns to include normalised RMSE

v1.4.1(2020-09-17) - F_o and F_m errors are now reported as percentages. - Model bias is now normalised. - Additional statistical output included in the form of normalised RMSE, where RMSE is normalised to the mean fluorescence level.

v1.4(2020-07-01)

- Added an option to ETR fitting to apply Serodio corrections for samples that have dark relaxation.
- Added an option for the fit to return E_k and manually calculate ETR^{max}.
- Added an option to fit ETR^{max} using a beta model.

v1.3.5(2020-05-22)

- Bug fix for ETR fitting when the number of measurements per light level is 1.

v1.3.4(2020-05-05)

- Bug fix for fitting with no connectivity model.

v1.3.3(2020-04-17)

- Bug fix for handling single FIRe files
- Loading FIRe files can now split single turnover and multiple turnover measurements

v1.3.2(2020-03-26)

- Bug fix to tools functions for change in name from fyield to flevel
- Bug fix for spectral correction to sort wavelengths to always be ascending

v1.3.1(2020-03-05)

- Added functionality to load raw data files of Single Acquisitions from Chelsea FastAct1 laboratory system
- Added functionality to calculate ETR by only using the last 3 measurements of each time step

v1.3(2020-02-20)

- Added functionality to load raw data files from Soliense LIFT-FRR
- Added functionality to load raw data files from Chelsea FastAct2 laboratory systems
- Bug fix to fit triple relaxation

v1.2.1(2020-01-08)

- Bug fix for fixed ro calculation

v1.2.1(2020-01-08)

- Bug fixes for fitting triple relaxation error codes
- Update to recommend bounds for tau1

v1.2(2019-12-09)

- Update to bias calculation.
- Update to spectral correction code for correcting for background light and using chlorophyll to calculate Kd.
- Update to spectral correction code to include FastOcean LED spectra.
- Update to spectral correction code that allows the user to include their own constants/spectra instead of the pre-included file.
- Plot functions now close any existing figure objects.
- Plot functions now include RMSE in the legend.
- Update to remove outliers code to make the datetime array 'datetime64'.
- Statistical metrics returned from FLC fitting procedure no longer include R^2, Chi^2 or reduced Chi^2.

v1.1(2019-10-15)

- Statistical metrics returned from fitting procedure no longer include R^2, Chi^2 or reduced Chi^2.

v1.0.2(2019-10-06)

- Saturation flashlets in relaxation fitting are now included in F:sub:'m'Relax estimation, rather than replacing relaxation flashlets.
- FIRe instrument relaxation bias now only uses the difference in relaxation flashlets to correct the large difference in flashlets.

v1.0.1(2019-10-04)

- Implementation of code for submission to PyPi. Package now available for installation using pip install phyto_photo_utils.

v1.0 (2019-10-01)

- Syntax changes to saturation, relaxation and flc. Different models now called with optional arguments instead of separate functions.

v0.9 (2019-10-01)

- Update to phytoplankton specific absorption code for handling phycobilin content
- Update to phytoplankton specific absorption code for updated pathlength amplification coefficients

- Update to phytoplankton specific absorption code for not normalising in the infra-red (750 nm) region

v0.8 (2019-06-28)

- Bug fix to spectral correction for handling arrays
- Statistical metrics now outputs RMSE, reduced Chi squared
- Processing flags now included in output

v0.7 (2019-06-20)

- F_o and F_m now estimated as intercepts of Huber Regression linear fits
- Fitting skipped if F_o is greater than F_m
- Spectral correction now calculates factor as a function of depth

v0.6 (2019-05-30)

- read the docs formatting applied
- added warning messages when lower bounds are higher than upper bounds
- added demo file

v0.5 (2019-05-23)

- various bug fixes
- spectral LED correction now estimates in situ light field

v0.4 (2019-05-21)

- added plot function

v0.3 (2019-05-17)

- restructured package to avoid nested functions
- added outlier removal tool to FLC function

v0.2 (2018-12-07)

- added functionality for FLCs

v0.1 (2018-12-01)

- Functions compiled in package format

CHAPTER 12

Contributing

When contributing to this repository, please first discuss the change you wish to make via issue, email, or any other method with the owners of this repository before making a change.

Please note we have a code of conduct, please follow it in all your interactions with the project.

12.1 Pull Request Process

1. Ensure any install or build dependencies are removed before the end of the layer when doing a build.
2. Update the README.md with details of changes to the interface, this includes new environment variables, exposed ports, useful file locations and container parameters.
3. Increase the version numbers in any examples files and the README.md to the new version that this Pull Request would represent. The versioning scheme we use is [SemVer](http://semver.org/).
4. You may merge the Pull Request in once you have the sign-off of two other developers, or if you do not have permission to do that, you may request the second reviewer to merge it for you.

12.2 Code of Conduct

12.2.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

12.2.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

12.2.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

12.2.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

12.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at tjryankeogh@gmail.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

12.2.6 Attribution

This Code of Conduct is adapted from the [Contributor Covenant][homepage], version 1.4, available at [http://contributor-covenant.org/version/1/4{]}{[}version]

[homepage]: http://contributor-covenant.org [version]: http://contributor-covenant.org/version/1/4/

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

phyto_photo_utils._etr, 39
phyto_photo_utils._load, 27
phyto_photo_utils._plot, 42
phyto_photo_utils._relaxation, 34
phyto_photo_utils._saturation, 32
phyto_photo_utils._spectral_correction, 37
phyto_photo_utils._tools, 31

Index

C

calculate_amplitude_etr() (*in module phyto_photo_utils._etr*), 39
calculate_blank_FastOcean() (*in module phyto_photo_utils._tools*), 31
calculate_blank_FIRe() (*in module phyto_photo_utils._tools*), 31
calculate_chl_specific_absorption() (*in module phyto_photo_utils._spectral_correction*), 37
calculate_instrument_led_correction() (*in module phyto_photo_utils._spectral_correction*), 38
correct_fire_instrument_bias() (*in module phyto_photo_utils._tools*), 31

F

fit_relaxation() (*in module phyto_photo_utils._relaxation*), 34
fit_saturation() (*in module phyto_photo_utils._saturation*), 32

L

load_FastOcean_files() (*in module phyto_photo_utils._load*), 29
load_FASTTrackaI_files() (*in module phyto_photo_utils._load*), 27
load_FIRe_files() (*in module phyto_photo_utils._load*), 28
load_LIFT_FRR_files() (*in module phyto_photo_utils._load*), 30

P

phyto_photo_utils._etr (*module*), 39
phyto_photo_utils._load (*module*), 27
phyto_photo_utils._plot (*module*), 42
phyto_photo_utils._relaxation (*module*), 34
phyto_photo_utils._saturation (*module*), 32
phyto_photo_utils._spectral_correction (*module*), 37
phyto_photo_utils._tools (*module*), 31
plot_fluorescence_light_curve() (*in module phyto_photo_utils._plot*), 42
plot_relaxation_data() (*in module phyto_photo_utils._plot*), 42
plot_saturation_data() (*in module phyto_photo_utils._plot*), 43

R

remove_outlier_from_time_average() (*in module phyto_photo_utils._tools*), 32