ECO$_2$S

A Quick Introduction and User Reference

by
Gerardo Fratini and Antonio Forgione

01 June 2010

* Since June 13th 2018
added the license
3-clause BSD as option

Prologue: The ECO$_2$S Project

The Eddy Covariance Community Software (ECO$_2$S) is free, open-source software, released under the GNU-GPL license.*The ECO$_2$S project started in 2005 from the recognition that the micro-meteorologist community involved in Eddy Covariance (EC) flux measurements was (and still is) far from reaching a satisfying agreement on a unique processing scheme of EC raw data to calculate fluxes, to be provided to carbon databases. Indeed, derivation of CO_2, H_2O and energy fluxes starting from raw wind, temperature and gas concentration data by means of the EC technique implies a remarkably long sequence of operations, including calibrations, corrections and statistical tests for assessing data quality or even to correct the raw data. However, the correct application, order, meaning and even the necessity of several processing steps are still topics under discussion.
As a result, EC processing software available to the community feature different implementations [4]: some operations may not be supported by some of the software, while some may be implemented in different ways even when they reflect the same conceptual assumptions. In addition, many research groups use custom software that may include further corrections suggested by recent findings. However, different data treatments may largely affect the consistency and inter-comparability of flux data produced by different groups and should be taken into account for a confident use of eddy flux data on the regional/continental scale. Furthermore, recently the number of EC stations all over the world is dramatically increased and is thought to increase even more in the next few years (see the Fluxnet website for an overview of existing sites). EC is becoming a standard *de facto* for measuring gas exchanges between the atmosphere and the biosphere, although it has not been standardized itself yet. As a consequence, nowadays many EC stations are run by ecology scientists or by technicians, rather than by micro-meteorologists. Thus, the potential inability of those in charge of providing flux data to correctly interpret results may further increase spatial and temporal inconsistencies of carbon databases.
With this in mind, a proposal was formulated by the University of Tuscia (Viterbo, Italy) to start a new software project, with the threefold aim to: *i.* provide the EC community with a complete, free, open-source tool to acquire and process data in a uniform fashion; *ii.* use this software to assess the statistical uncertainties related to different EC processing schemes and stick it to EC results and *iii.* give the community the possibility to re-process old datasets, to account for recent findings and refinements to the methods.
But, why a new software?
The main reason for starting a new EC software project is that none of the existing software projects is intended to be both freely distributed and open-source. Especially the latter point is of utmost importance for a number of reasons, including:

- an open-source EC software can become a platform on which new solutions can be implemented and compared against previous alternatives, while assuring a stable, shared and agreed playground;
- new developments around EC, also beyond standard processing, can be easily, quickly and effectively made available to the community, without a need for developing a new user environment each time. This would be the case, for example, for complex footprint models, uncertainty estimations, off-line spectral correction procedures, refinements to tilt correction or flow distortion compensation algorithms, among the rest;
- the possibility to access the source code can make any user confident on what he is actually doing to his data throughout the processing. Possibly, it can also help raising user consciousness on the EC data treatment;
- many different raw data file types exist in the community. To reprocess them in a uniform and standardized fashion, they all need to be converted into a unique file format. This is a huge task that however may be easily afforded by using an open-source platform, with the short-term contribution of a number of developers.

In order to be time-effective and successful, the project must rely on the active contribution of all those in the community who are involved in the continuous refinement of EC methodology and interpretations as well as, of course, a few programmers. Although the project started in the framework of the CarboEuropre-IP project and is presently partly supported by the IMECC project, as a matter of fact this is an independent, mostly self-sustained, transverse action, linked to no specific project deliverable but yet of potential interest for all the projects involving EC measurements, especially those where standardisation and representativeness of large-scale datasets are targeted.

Introduction

This document is a quick introduction to the use of ECO$_2$S, the Eddy COvariance COmmunity Software. ECO$_2$S is a suite of data acquisition and processing tools, used to calculate Eddy Covariance fluxes and a number of statistics and quality assessments of atmospheric turbulence data. The present release (ECO$_2$S 1.0.0) is the first official version, released specifically for being tested by a limited number of users. The suite is developed by and belongs to the "Eddy Covariance Community", a general expression used to gather together all those who continuously contribute to the development and refinement of the Eddy Covariance methodology, by means of scientific publications and development of processing routines.
Although ECO$_2$S is developed to be used both through its Graphical Users Interface (GUI) and in "command line" mode, using the GUI is strongly recommended. In fact, far from being just an interface to the processing engines, the GUI is an intelligent application that drives the users throughout the processing steps, avoiding most of the incompatible or unsuitable options, which are inherently possible when the processing goes through a series of, at least, two steps. The present document introduces you to the use of ECO$_2$S with its Graphical Interface.

1 Installing ECO$_2$S

The current version of ECO$_2$S is released for Microsoft Windows operating systems. It is known to work properly under Windows XP/Vista (including 64 bit versions) and supposed to work also under Windows 7 and former versions, such as Windows 2000/NT. Linux and Mac OS releases can be compiled on demand, although they are not continuously tested, thus proper functioning is not assured. Installing ECO$_2$S under Windows is as easy as it is for any other modern program. Locate the downloaded installer, double-click and follow the instructions. A series of *next* may well fit your installation intentions.

2 ECO$_2$S::Logging and ECO$_2$S::Processing

ECO$_2$S software package is intended to be comprised of two main programs. ECO$_2$S::Logging shall be used to acquire data from an Eddy Covariance station, while ECO$_2$S::Processing is used to process those data. Presently, ECO$_2$S::Logging is not still embedded in the ECO$_2$S graphical interface, but is available as a separate tool, with the former name of ECO$_2$Catch. This acquisition software collects raw data in a new format, that is described in details in Appendix A. Because the extension of those raw files is ".ene", in the following we will refer to them as ENE-files. Briefly, an ENE-file is an archive containing the data file in plain text, with comma-separated values, and a configuration file, containing acquisition-time information that are useful when processing the paired data file. Sample ENE-files are distributed along with this document. ECO$_2$S::Processing, the processing software, is specifically developed to work on raw files in ENE format. However, a number of other raw formats is also supported, and a tool provided to simulate the ENE processing.

3 Overview of ECO$_2$S::Processing structure

In Figure 1 a schematic diagram provides an overview of ECO$_2$S processors, main input/output and dependencies. Processing raw data with ECO$_2$S to obtain fluxes involves a minimum of two steps. In the first step, raw data are processed by ECO$_2$S::PreProcessor and all calculations that can/must be done at raw data level are performed. To properly process raw data, ECO$_2$S::PreProcessor needs ancillary information, provided in the ECO$_2$S::Logging project file. If raw files have been acquired with ECO$_2$Catch, these information are stored inside the ENE-files and are similarly available in the acquisition project file. If raw files were collected with another software, the ECO$_2$S::Logging project file must be compiled by the user (see following Sect. 4.1)

ECO$_2$S::PreProcessor outputs a number of partial or final results, used for further processing or user analysis. Among them[1]:

- main statistics on raw files, such as mean values, (co-)variances, standard deviations, skewnesses and Kurtosis, at 7 different levels of processing;
- results (flags) of a screening procedure for statistical data quality;
- on demand, all relevant (co-)spectra for each raw file;
- an estimation of main meteorological variables (air temperature, relative humidity and pressure), based on raw data and site characteristics;
- calculated time-lags and rotation angles for tilt correction;
- results of the so-called "stationarity test", used later for flagging final fluxes for quality;
- tentative flux estimations, used as inputs for other processing steps.

If a thorough spectral analysis is proved necessary for your EC station (e.g. in view of significant spectral corrections), you might conveniently used ECO$_2$S::SpectralAnalysis. Basing on ECO$_2$S::PreProcessor outputs,

[1]Other outputs are also provided, not detailed here.

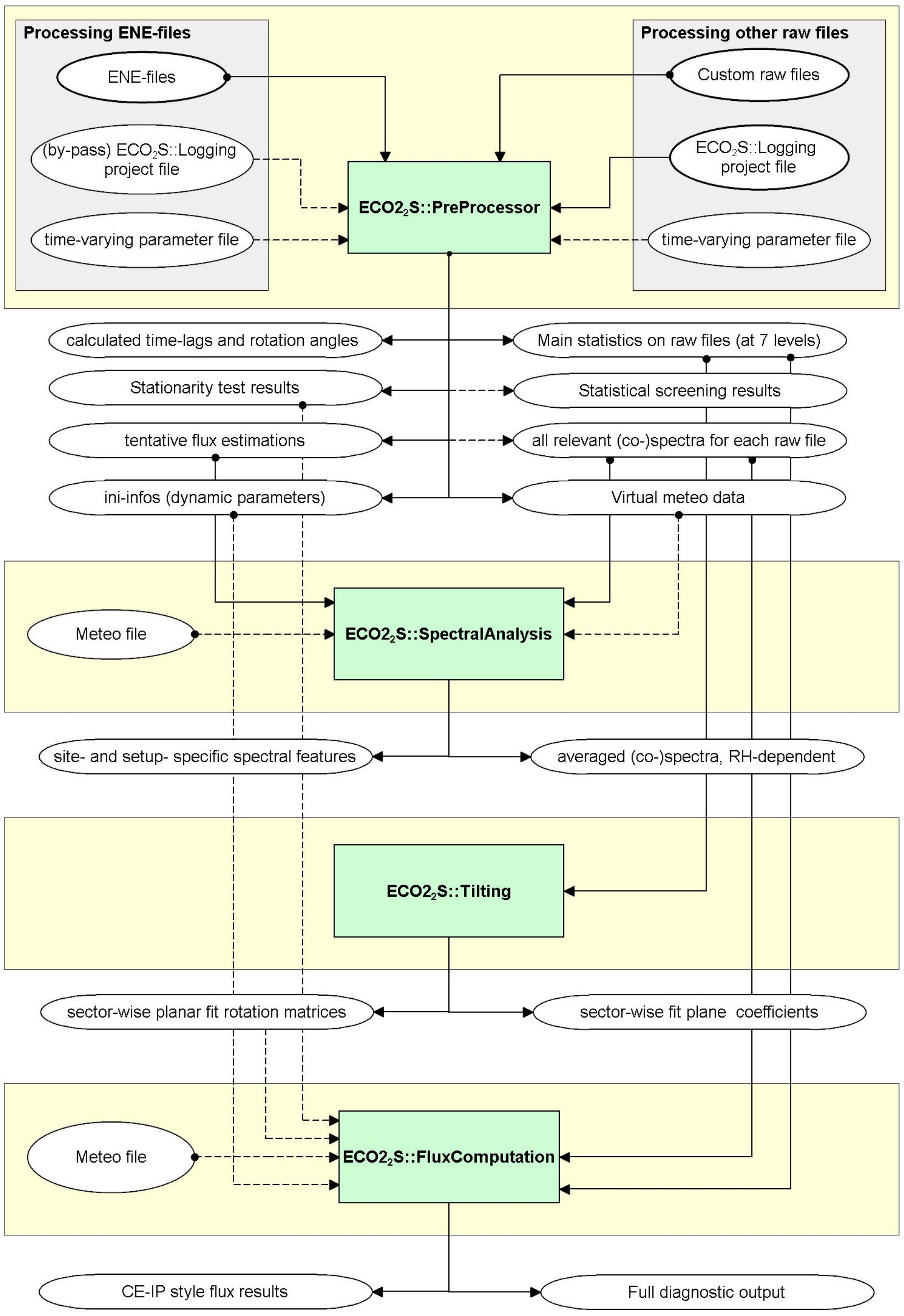

Processing ENE-files
ENE-files
(by-pass) ECO$_2$S::Logging project file
time-varying parameter file
Processing other raw files
Custom raw files
ECO$_2$S::Logging project file
time-varying parameter file
ECO2$_2$S::PreProcessor
calculated time-lags and rotation angles
Main statistics on raw files (at 7 levels)
Stationarity test results
Statistical screening results
tentative flux estimations
all relevant (co-)spectra for each raw file
ini-infos (dynamic parameters)
Virtual meteo data
Meteo file
ECO2$_2$S::SpectralAnalysis
site- and setup- specific spectral features
averaged (co-)spectra, RH-dependent
ECO2$_2$S::Tilting
sector-wise planar fit rotation matrices
sector-wise fit plane coefficients
Meteo file
ECO2$_2$S::FluxComputation
CE-IP style flux results
Full diagnostic output

Figure 1: Block diagram of ECO$_2$S structure

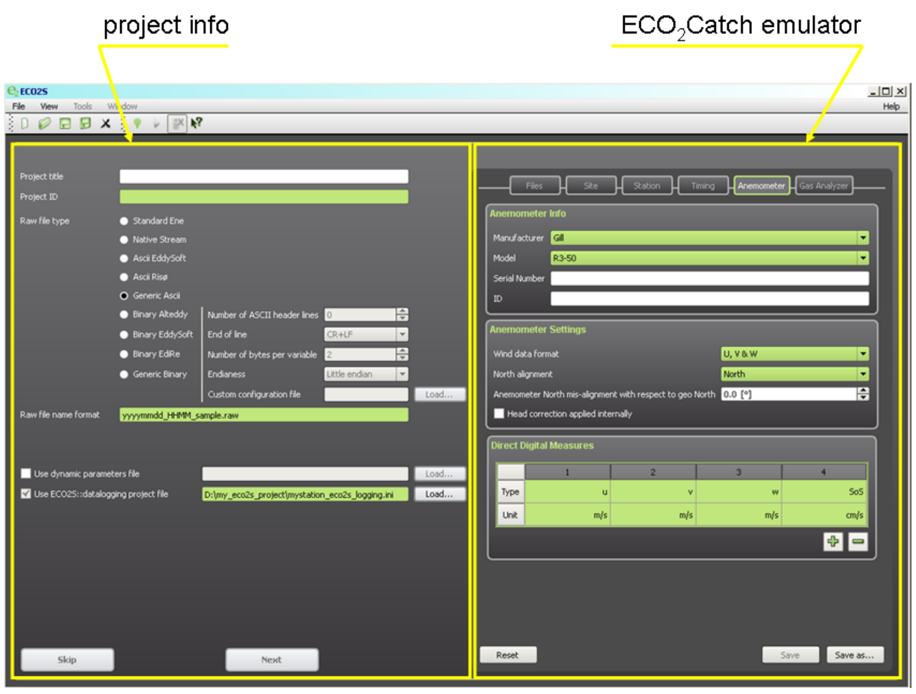
ECO$_2$S::SpectralAnalysis calculates average w-cospectra of sonic temperature and CO_2 and H_2O concentrations; it performs an analysis of $w - H_2O$ cospectra dependency on relative humidity and determine RH-dependent cut-off frequencies of the low-pass transfer functions representing the filtering properties of the EC system.
Similarly, if your site is such that the (sector-wise) planar-fit method for tilt correction is advisable, you shall use ECO$_2$S::Tilting to calculate fitting plane coefficients and related rotation matrices.
Finally, ECO$_2$S::FluxComputation is the tool you need to calculate corrected fluxes. It starts from the last most elaborate statistics file (level 7) and calculates corrected fluxes and quality flags. Depending on your processing choices, you may need to provide result files obtained with ECO$_2$S::Tilting and/or ECO$_2$S::SpectralAnalysis. ECO$_2$S::FluxComputation outputs two files: *i.* a comprehensive output file, containing many information besides corrected fluxes, such as un-corrected fluxes, average gas concentrations and densities, average wind speed, turbulence parameters (u^*, T^*, stability parameter), footprint estimations and storage estimations among the others; *ii.* a reduced output file, containing corrected fluxes, quality flags, mean concentrations and wind speed. This file is formatted as to be directly submitted to the Fluxnet database (Dario???).

4 Running a processing session with ECO$_2$S::Processing

4.1 Compiling the Project window

Double-clicking on ECO$_2$S icon, the welcome message prompts you to choose whether you want to acquire Eddy Covariance data or to process data you already have. Click on "::Data Processing" to enter ECO$_2$S::Processing. Processing sessions, similarly to acquisition sessions, are organised as *processing projects*, that can be created, saved and opened by doing so on the corresponding project file. For these operations, just use the toolbar or the menus in the usual way. Project files are text files structured in INI format, containing all information entered and processing options selected by the user on the current project. Note that ECO$_2$S::Processing can handle multiple projects at one time.

Before entering the real processing interface, a preliminary screen, the Project window, asks to enter information on the current project: *i.* choose a title; *ii.* choose a (mandatory) ID, that will be attached to all outputs for identification (as a general rule of ECO$_2$S interface, mandatory fields are evidenced in green); *iii.* select the raw files type. If your raw file type is not listed don't worry, because it is likely to be equivalent to one of those available (ask us for more information).



Project title:
Project ID:
Raw file type: ● Standard Ene ● Native Stream ● Ascii EddySoft ● Ascii Risø [✓ Generic Ascii] ● Binary Alteddy ● Binary EddySoft ● Binary EdiRe ● Generic Binary

Number of ASCII header lines: 0
End of line: CR+LF
Number of bytes per variable: 2
Endianess: Little endian
Custom configuration file:

Raw file name format: yyyymmdd_HHMM_sample.raw

☐ Use dynamic parameters file
☑ Use ECO2S::datalogging project file: D:\my_eco2s_project\mystation_eco2s_logging.ini

Anemometer Info
Manufacturer: Gill
Model: R3-50
Serial Number:
ID:

Anemometer Settings
Wind data format: U, V & W
North alignment: North
Anemometer North mis-alignment with respect to geo North: 0.0 [°]
☐ Head correction applied internally

Direct Digital Measures

	1	2	3	4
Type	u	v	w	SoS
Unit	m/s	m/s	m/s	cm/s

Figure 2: Snapshot of ECO$_2$S project window

Raw file types supported are:

- `Standard Ene`, ECO$_2$S raw data format (see Appendix A);
- `Native Streams`, native anemometer outputs (see Appendix A);
- `ASCII EddySoft`, comma-separated data files, as converted from EddySoft native binaries, by means of EddyRead;
- `ASCII Risø`, a customised text file type;
- `Generic ASCII`, generic text files, with data columns separated by either commas, semi-columns, spaces, tabs etc.;
- `Binary Alteddy`, binary files as produced by Alteddy;
- `Binary EddySoft`, binary files (`.slt`) as produced by EddySoft, featuring a customised binary header;
- `Binary EdiRe`, binary files (`.slt`) as produced by EdiRE, featuring a customised binary header;
- `GenericBinary`, a generic binary format.

Binary formats can be further specified, by entering the number of ASCII header lines (if applicable), the line terminator, the number of bytes per variable, and the endianess.
If you selected binary `Binary EddySoft` or `Binary EdiRe`, you are now asked to enter another file (`custom configuration file`), needed to interpret the binary format. This is a standard configuration (extension `.cfg`) file in the case of EddySoft and a processing list in the case of EdiRe.

If you want to process ENE files in the standard way, your job here is finished and you can `skip` to the next phase (Figure 3, Sol. 1) . Otherwise, select the relevant data type. In this case, the right side of the GUI will activate: this is the ECO_2S::Logging emulator: here, you must enter all the information relevant to the site, station, setup used to acquire the data you are going to process. This interface is basically that of ECO_2S::Logging, and it is pretty much the same as the latest interface of ECO_2Catch (ver.0.9.8). Refer to Appendix B for a brief introduction to ECO_2Catch. Note that, once you have entered all mandatory fields here, you must save the corresponding ECO_2S::Logging project file, and this will appear in the `ECO`$_2$`S::datalogging project` entry (Figure 3, Sol. 4). Alternatively, if you already have an ECO_2S::Logging project file, you can load it and then check on the right side that all information have been properly imported and modify as needed.

In case of ENE files, these information are stored in the configuration file zipped in the archive, that's why you do not need to enter them again here. However, even when processing ENE files, you might want to by-pass settings selected at acquisition time, by using an independent ECO_2S::Logging project file. This could be convenient especially in two occasions:

- when ancillary information stored in ENE files must be ignored because incorrect or incomplete (Figure 3, Sol. 3);
- when all ENE files contain the same ancillary information, that can thus be read only once, to speed up the data processing operation (Figure 3, Sol. 2).

In this cases, the easiest way to go is to unzip one ENE archive, detect the INI file contained in it (most likely called *eco2s_processing.ini*), load it from the `ECO2S::datalogging project` entry, modify it as needed and save. This will later be used instead of the configuration files residing inside the ENE files.
If you are not using ENE files, you also must enter the `Raw file name format`: this is a prototype file name, used to correctly interpret file names to retrieve date/time information. The easiest way to enter the prototype is to copy and paste here the name of any raw file and then substitute the date/time information with the following hot keys:

- *yy* for a 2-digits year

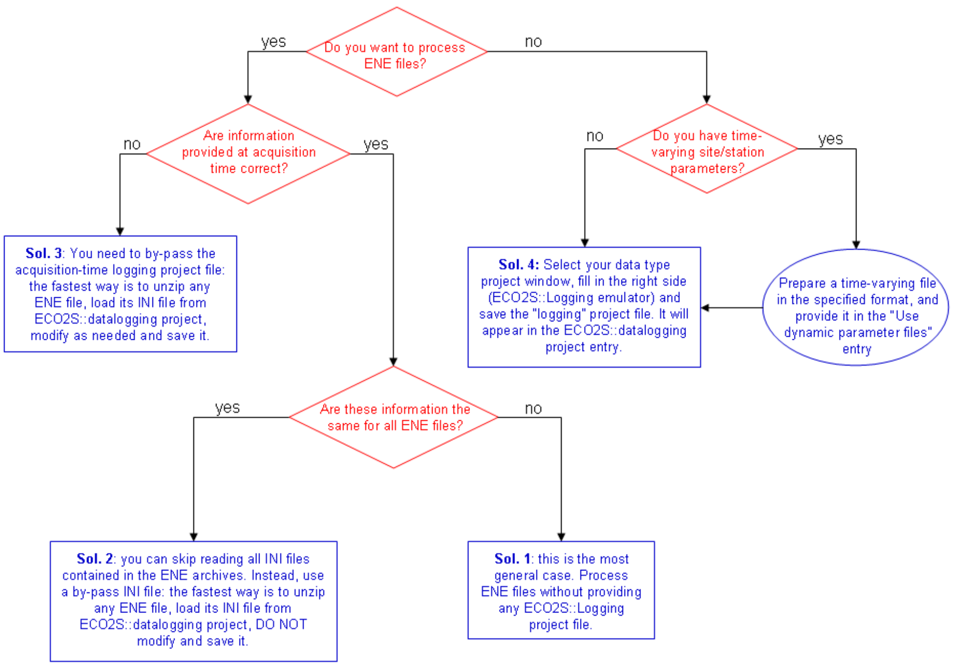

Do you want to process ENE files?
yes
no
Are information provided at acquisition time correct?
no
yes
Do you have time-varying site/station parameters?
no
yes
Sol. 3: You need to by-pass the acquisition-time logging project file: the fastest way is to unzip any ENE file, load its INI file from ECO2S::datalogging project, modify as needed and save it.
Sol. 4: Select your data type project window, fill in the right side (ECO2S::Logging emulator) and save the "logging" project file. It will appear in the ECO2S::datalogging project entry.
Prepare a time-varying file in the specified format, and provide it in the "Use dynamic parameter files" entry
Are these information the same for all ENE files?
yes
no
Sol. 2: you can skip reading all INI files contained in the ENE archives. Instead, use a by-pass INI file: the fastest way is to unzip any ENE file, load its INI file from ECO2S::datalogging project, DO NOT modify and save it.
Sol. 1: this is the most general case. Process ENE files without providing any ECO2S::Logging project file.

Figure 3: Decision tree at ECO$_2$S project window, depending on raw file type and availability of time-varying parameters information

- *yyyy* for a 4-digits year
- *dd* for a 2-digits day-of-month
- *mm* for a 2-digits month-of-the-year
- *ddd* for a 3-digits day-of-year (DOY)
- *HH* for a 2-digits hour
- *MM* for a 2-digits minute

Please note that: *i.* All date/time information must be present in file names. While day/month (or DOY) and hour information are virtually always available, years and minutes are sometimes not reported on file names. For these cases, a tool is available (on demand), that allows automatic renaming files, introducing missing fields; *ii.* as for this release, only DOY/4-digits year and day-of-month/month-of-the-year/2-digits year pairs are not supported; *iii.* hot keys stored in the extension are not supported [2]; *iv.* any non sensitive

[2]In this case, as a preliminary step, rename files including information missing in the file name.

character shall be left as it is, with the exception of sub-strings that are equal to any of the above hot keys, in this case just substitute it with any other sub-string of the same length; *v.* the extension must be included in the prototype. As an example, if the sample raw file name is:

`mysite_2009_001_0130.dat`

acceptable prototypes would be:

`mysite_yyyy_ddd_HHMM.dat`, or `xxxxxx_yyyy_ddd_HHMM.dat`;

but if the prototype is

`mysite_0021200.R08`

then information about the year ('08' in the extension) cannot be retrieved, thus files shall be first renamed, e.g. like:

`mysite_080021200.R08`

so that a prototype could be:

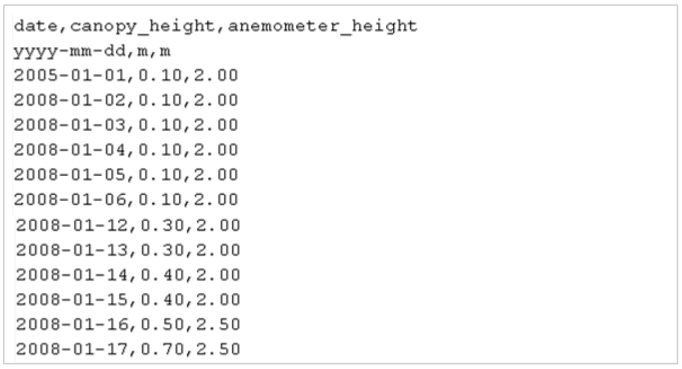
`mysite_yydddHHMM.R08`.

In the following, examples of unsupported file prototypes are listed:

- `ddmmyy_HHMM_mysite.raw` (2-digits years not allowed with 'mm' 'dd' hot keys)
- `mysite_dddyyyy_HHMM.raw` (4-digits year not allowed with DOY)
- `ddmmyyyy_HH_mysite.raw` (minute information is missing)

Finally, an entry is available for providing time-varying site/station information. This option might be useful when processing files other than ENE. In fact, in the `ECO2S::datalogging project` you were asked to enter site/station/setup information that are intended to be identical for all raw files. However, there might be cases when some parameters are time-dependent, thus these should not be set equal for all files. For example, think about the canopy height - and possibly the anemometer height - of a cropland site, affecting the estimation of the stability parameter $(z - d)/L$ and the spectral features of the time series. In order to properly take into account time-dependent parameters, before starting the processing procedure the user shall prepare a text file containing such time-varying values. Currently, only a subset of possible parameters can be provided this way, namely:

- The canopy height
- The anemometer height (or measurement height)



```
date,canopy_height,anemometer_height
yyyy-mm-dd,m,m
2005-01-01,0.10,2.00
2008-01-02,0.10,2.00
2008-01-03,0.10,2.00
2008-01-04,0.10,2.00
2008-01-05,0.10,2.00
2008-01-06,0.10,2.00
2008-01-12,0.30,2.00
2008-01-13,0.30,2.00
2008-01-14,0.40,2.00
2008-01-15,0.40,2.00
2008-01-16,0.50,2.50
2008-01-17,0.70,2.50
```

Figure 4: Sample of a time-varying (dynamic) parameters file, containing daily values of canopy height and anemometer height.

- The zero-plane displace height
- The canopy roughness length
- The anemometer misalignment with respect to geographic North

The format of this file is rigorously defined: dates and times must be entered in ISO format (yyyy-mm-dd and HH:MM respectively); variable names and units are predefined (see Table 1); values must be separated by commas. On the other hand, large flexibility is provided: time information can include either only dates, only times or both dates and times; only the parameters of interested shall be included, placed in any order (but dates and times first); finally, data can be provided on a continuous time basis (e.g. for all days of the year, with repeated values if necessary), or only on selected dates/times, thereby assuming that parameters remain constant until the next provided date/time.

variable	**label**	**units**
date	date	yyyy-mm-dd
time	time	hh:mm
canopy height	canopy_height	m
anemometer height	anemometer_height	m
zero-plane displacement height	displacement_height	m
canopy roughness length	roughness_length	m
sonic North misalignment	north_offset	°

Table 1: Labels and units of time-varying (dynamic) parameters.

Once all the necessary information have been provided in the project window, the `next` button activates, that takes you to the interface of the processing engines.

4.2 Processing raw files with ECO_2S::PreProcessor

Processing raw data to get Eddy Covariance fluxes involves a remarkably long sequence of operations, such as calibrations, corrections and quality assessments. Some of these operations must be done at *raw data level*, while others can be calculated from suitable statistics derived from the raw data. Accordingly, the processing procedure in ECO_2S is split in (at least) two steps. The first, preliminary step is done with ECO_2S::PreProcessor and consists in importing the raw data and perform all *raw data level* operations. All other calculations (e.g. spectral analysis, planar fit, flux computation) are done in a second step, and rely on the results of ECO_2S::PreProcessor. This approach was adopted for two main reasons: *i.* raw-level operations require a long time to be performed (about 12 hours for 1 year raw data, on a relatively modern laptop), thus you might find convenient to do them only once, while different post-processing trials can be conveniently performed in short time; *ii.* results of the pre-processor are in a standard format, regardless of the raw data type considered, thus future additions to the software suite do not have to cope with problems related to different raw file types.
ECO_2S::PreProcessor performs all operations needed at raw data level, in particular:

- reads raw files and configuration files;
- extracts the proper dataset, according to the chosen flux averaging period; if the case, splits raw files in sub-periods or merges consecutive files for longer averaging times;
- performs statistical tests on the raw dataset, according to guidelines provided in [6]; if the case, eliminates spikes or single data values out of plausible ranges.
- applies the sample-wise cross-wind correction (implemented after [3]) to sonic temperature, if the case;
- applies an angle-of-attack correction (implemented after [5]) if requested;
- converts molar fractions into dry mixing ratios sample by sample, if needed;
- performs the ”2D rotations” tilt correction (implemented after [7]) if requested;

- de-trends time series (either by block averaging, linear detrending, exponentially-weighted averaging or by running mean);
- determines scalar time lags and compensate them if requested;
- calculates tentative fluxes and estimates average values of key meteorological variables, needed in the following processing steps (i.e., air temperature, pressure, relative humidity, water vapour actual and saturation partial pressure, air molar volume, vapour pressure deficit, dew-point temperature)
- calculates auxiliary values, needed for later spectral analysis (statistics on *digitally degraded temperature time series*, not further described here).
- calculates all relevant spectra and co-spectra for each averaging period.
- at the very beginning and after each of the above processing steps, calculates main statistics on the time series (mean, standard deviation, (co-)variance, skewness, kurtosis).

Entering in ECO$_2$S::Processing, the graphical interface defaults on ECO$_2$S::PreProcessor $\gg$ `General Options`. You can always go back here from any position in the GUI, by selecting the `Raw Processing` icon on the programs menu on the left side.

4.2.1 The `General Options` tab

Enter here the raw data directory (make sure that the files contained in this folder and having your raw-file extension, are only actual raw files!) and the main output directory, where ECO$_2$S will create a set of sub-folders, containing different outputs.
Then, the averaging interval must be chosen. If you want the averaging time to be the same as the files length, choose '0' as the `flux averaging interval`. If the flux averaging time is longer than the raw file duration, you can instruct ECO$_2$S::PreProcessor to merge a suitable number of consecutive files before picking the proper averaging dataset. Consider raw file length and flux averaging time to decide how best to merge raw files: for example, if your files are 30 minutes long and you want fluxes averaged over 45 minutes, consider merging 3 raw files (=90 minutes), from which 2 flux values (2 x 45 minutes) will be calculated. The `Timestamp tolerance` allows you to adjust the time information of spurious raw files to the rest of the dataset. As an example, consider a flux dataset built up using 30 minutes averaging intervals and starting at a round hour, e.g. 8:00; in this case, the following times should be 8:30, 9:00, 9:30 etc. Now, it sometimes happens to have files starting/ending a few minutes away from a round time, or analogously

ending before a round time (e.g. 8:02, 8:56 etc..). Setting a timestamp tolerance of, for example, 5 minutes, will cause ECO_2S to adjust time stamps that are within ±5 minutes from the closest round time, so as to obtain an ordered dataset. As an example, given the set of file names:

```
mysite_20100601_0800.dat

mysite_20100601_0830.dat

mysite_20100601_0902.dat

mysite_20100601_0930.dat

mysite_20100601_0959.dat

mysite_20100601_1030.dat

mysite_20100601_1106.dat
```

setting a tolerance of 5 minutes will cause the dataset to have the following dates/times:

```
2010-06-01,08:00

2010-06-01,08:30

2010-06-01,09:00

2010-06-01,09:30

2010-06-01,10:00

2010-06-01,10:30

2010-06-01,11:06
```

where the last time as not been adjusted, because its delay from `11:00` is larger than 5 minutes. Finally, checking the `Create continuous dataset` will automatically fill all outputs with lines corresponding to missing files, providing you with a continuous dataset, in terms of dates and times, where of course missing values are replaced with error codes (-9999.0 everywhere in ECO_2S).

In the `Settings` section you must now choose your processing options. ECO_2S proposes default values for each option, which however should by no means be considered as "best choices", the latter depending strictly on your combination of site characteristics and EC set-up. Hereafter, a quick description of all processing option is provided:

- `Maximum accepted percentage of missing lines per file`: allows you to specify which amount of missing or invalid data samples is to be accepted; file(s) missing a larger amount of samples will be skipped by the software.
- `Wind speed measurement off-sets`: allows you to account for any known bias in your anemometer's wind measurements (to assess such possible off-sets, you may enclose the anemometer in a box and consider any deviation from zero in wind component determination).
- Optional `Cross-wind correction for sonic temperature`: implemented after [3], to be applied if it is not performed internally in your anemometer unit. This correction is available for the following sonics: HS-50, HS-100, R2, R3-50, R3-100, R3A-100 by Gill, USA-1 by Metek, CSAT3 by Campbell Scientific, 81000 by Young.
- `Angle-of-attack correction`: implemented after [5], to correct for sonic head disturbances introduced at large wind attack angles. This correction is currently available for the following sonics: WindMaster, WindMaster Pro, R2, R3 and R3-50, all by Gill.
- `axis rotation for tilt correction`: options available (all implemented after [7]) are:
  1. sample-wise 2D rotations, *before detrending*;
  2. 2D rotations performed on statistics, *after detrending*;
  3. 3D rotations;
  4. sector-wise planar-fit.

  Options 2 to 4 are not performed by ECO$_2$S::PreProcessor, rather these are applied at the flux computation level. Furthermore, to apply the planar fit option, you will need to run ECO$_2$S::Tilting before running ECO$_2$S::FluxComputation.
- `detrend method`: low frequency cut-off options are:
  1. block (Reynolds) average;
  2. linear detrending;
  3. running mean;
  4. exponentially-weighted average.

  Options 3 and 4 require a suitable time constant to be provided.
- `time-lag compensation`: refers to the possible time misalignment between the measurement of simultaneous wind components and gas concentrations. Three options are currently available:

1. use of user-defined, fixed time-lags, detailed for each gaseous species;
2. detection, by means of covariance maximisation, of the most likely time-lag within a user-defined time window; if a best estimation is not attained within the window, user-supplied default values are used;
3. as in 2., but without defaults (the maximum covariance determines the time lag in all cases);

Shortly a new option will be available for closed-path systems, allowing to adapt the searching window and default values to the file-specific air relative humidity, to account for the RH-dependence of the transit time of air parcels in the sampling line[3].

The remaining options in the `Settings` section all refer to the Fourier analysis of raw files. On request, ECO$_2$S::PreProcessor can output:

1. full w-T co-spectrum for each raw file;
2. all relevant spectra and co-spectra, reduced by exponentially-spaced binning;

These outputs are designed to be imported by ECO$_2$S::SpectralAnalysis for the purpose of calculating spectral correction parameters. However, if you do not plan to perform a detailed spectral analysis/correction, you may consider avoiding outputting such files, because they take a considerable amount of time to be calculated and written on files; they also will occupy a not negligible amount of disk space (roughly as much as the raw files themselves). In order to perform the Fourier transform, time series must be filtered using a `Window for data tapering before FFT-ing`. Finally, the number of exponentially-spaced frequency bins for (co-)spectra reduction must be entered.

4.2.2 The `Statistical Tests` tab

In the `Statistical Tests` tab you can choose which test are to be performed on your raw files. Nine statistical tests, all based on [6], are available, namely:

1. spike detection (and removal) test;
2. amplitude resolution test;
3. drop-outs test;
4. absolute limits test;

[3]This option is already available in command-line mode.

5. skewness and kurtosis test;
6. discontinuities test;
7. time-lags test;
8. angle of attack test;
9. steadiness of horizontal wind test.

Refer to [6] for a description of the tests and for setting test parameters. ECO_2S::PreProcessor proposes default values for each selectable test parameter. However, as pointed out in [6], the proper setting depends on the site and EC setup under consideration.
Activating at least one test will cause ECO_2S to produce an output file with resulting flags for the selected tests. According to the authors, this information should be used to discard raw files for which "hard-flags" are detected. However, ECO_2S does not perform such a selection; it is left up to the user whether to discard results for flagged files or not. As a suggestion, the spike detection test shall always be performed, as it actually serves also to substitute spikes with a linear interpolation of neighbouring samples. Not eliminating spikes will most likely result in unrealistic flux estimations. Furthermore, by default all tests are activated, because in fact performing those tests does not increase the computing time dramatically.

4.2.3 Quick look at ECO_2S::Processing output files

ECO_2S::PreProcessor creates up to 5 sub-folders, inside the selected output directory, namely:

1. `eco2s_stats`;
2. `eco2s_aux`;
3. `eco2s_rawscreen`;
4. `eco2s_bin_cospectra`;
5. `eco2s_cospectra_wT`.

The first three folders are created (if they do not exist yet) at each run, while the last two are created only if the respective outputs (binned cospectra and full w-T cospectra) have been selected, see Sect. 4.2.1. In the first three folders, summary result files are stored. Each line of such files refers to a flux averaging interval and report, at the beginning, the raw file it refers to and the date/time of *the end* of the averaging interval. The name of these result files are created using the processing project ID entered by the user in the entry window (Sect. 4.1). On the contrary, the cospectra folders (4 and 5) contain one co-spectra (binned and full, respectively) file for each raw file.

The eco2s_stats *folder*
Contains main statistics (mean values, (co-)variances, skewness and kurtosis) derived from the time series and calculated at 7 different processing levels, namely: *i.* on the very raw data (file with suffix `_st1`); *ii.* after compensating for wind measurement off-sets (suffix `_st2`); *iii.* after applying the cross-wind correction (suffix `_st3`); *iv.* after applying the angle-of-attack correction (suffix `_st4`); *v.* after sample-wise 2D rotations (suffix `_st5`); *vi.* after detrending (suffix `_st6`); *vii.* after time-lags compensation (suffix `_st7`). Obviously, if any of the above processing steps are not applied, one or more statistics file may result identical.

The eco2s_aux *folder*
It contains a miscellanea of output files:

- `eco2s_projID_rawfluxes.csv` contains tentative estimations of most relevant fluxes;
- `eco2s_projID_timelags.csv` contains calculated time lags for main scalars;
- `eco2s_projID_virt_meteo.csv` contains an estimation of meteorological parameters (air temperature, barometric pressure and relative humidity), as derived from raw data and from site characteristics, along with estimations of water vapour actual and saturation partial pressures, VPD, air molar volume and dew-point temperature;
- `eco2s_projID_qc_test.csv` contains results of the *stationarity test*, implemented after [1]. This file must be provided to ECO_2S::FluxComputation, that performs the so-called *integral turbulence test* and derives quality flags, also according to [1];
- `eco2s_projID_ini_info.csv` contains time-varying parameters. These are derived either from the configuration files residing in the ENE archives or from the `Dynamic parameters file` provided in the Project window. This file is produced in any case, i.e. even if parameters are constant throughout the raw dataset. It can later be optionally used by ECO_2S::FluxComputation for retrieving time-varying parameters;
- `eco2s_projID_vdegT_cov.csv`, not described here, is an auxiliary file needed for *in situ* spectral analysis, to evaluate the effect of a closed-path EC sampling system as a low-pass filter for atmospheric scalar concentrations (see [2] for an introduction).

The eco2s_rawscreen *folder*
Currently, it contains two output files:

- `eco2s_projID_rs_flags.csv` contains summary results of nine statistical tests applied at the raw data level, according to [6]. In the header, a legend explains the symbols used in the file: single variables are flagged "1" if the test failed (that is, if the variable shows a statistical problem), "0" otherwise. Flags for tests not performed are set to "9". For some tests, "soft flags" (prefix `SF_`) and "hard flags" (prefix `HF_`) are available, the latter evidencing more serious problems. Variables are labeled as: u and v for the horizontal components of the wind vector, w for the vertical component, t for sonic temperature, c for CO_2 concentration, h for water vapour concentration, a for wind angle-of-attach and U for the module of horizontal wind speed. No unambiguous policy exists on how to use the flags to discard "bad data files". In [6], authors suggest to discard all file that show at least one hard flag, pointing out that, however, a definitive choice should follow from an in-depth investigation of each flagged raw file;

- `eco2s_projID_rs_spike_test.csv` contains more detailed information on the spike detection test, including the number of spiked detected (and eliminated) for each variable and the number of repetitions needed to eliminate all spikes in each file. Spikes are counted in two different way: "individual spikes" (`ind_spikes`) count the number of individual samples detected as spikes, while consecutive outlying samples are defined as a single "spike" and thse are counted separately (`spikes`). Variables labels intuitive.

The eco2s_bin_cospectra *folder* If the option for outputting binned-cospectra is checked, this folder *The* eco2s_cospectra_wT *folder*

4.3 Preparing planar-fit rotations with ECO$_2$S::Tilting

4.4 Preparing *in situ* ECO$_2$S::SpectralAnalysis

References

[1] T. Foken, M. Goeckede, M. Mauder, L. Mahrt, B.D. Amiro, and J.W. Munger. Post-field quality control. In X. Lee, W. J. Massman, and B. E. Law, editors, *Handbook of micrometeorology: a guide for surface flux measurements*, pages 81–108. Kluwer, Dordrecht, 2004.

[2] A. Ibrom, E. Dellwik, H. Flyvbjerg, N.O. Jensen, and K. Pilegaard. Strong low-pass filtering effects on water vaopur flux measurements with closed-path eddy correlation systems. *Agricultural and Forest Meteorology*, 147:140–156, 2007.

[3] H. Liu, G. Peters, and T. Foken. New equations for sonic temperature variance and buoyancy heat flux with an omnidirectional sonic anemometer. *Boundary-Layer Meteorology*, 100:459–468, 2001.

[4] M. Mauder, T. Foken, R. Clement, J.A. Elbers, W. Eugster, T. Gruenwald, B. Heusinkveld, and O. Kolle. Quality control of carboeurope flux data - part ii: Inter-comparison of eddy-covariance software. *Biogeosciences Discussions*, 4:4067–4069, 2007.

[5] T. Nakai, M.K. van der Molen, J.H.C. Gash, and Y. Kodama. Correction of sonic anemometer angle of attack errors. *Agricultural and Forest Meteorology*, 136:19–30, 2006.

[6] D. Vickers and L. Mahrt. Quality control and flux sampling problems for tower and aircraft data. *Journal of Atmospheric and Oceanic Technology*, 14:512–526, 1997.

[7] J.M. Wilczak, S.P. Oncley, and S.A. Stage. Sonic anemometer tilt correction algorithms. *Boundary-Layer Meteorology*, 99:127–150, 2001.

A ENE file format

ECO$_2$S proposes a new standard format of EC data and meta-data files. As a general rule, all data files (raw files, output result files, initialisation files etc..) are plain texts, editable with any text editor. Initialisation and configuration files are written in a simple INI format (however, not a standard one), while raw data files and all output files (with the exception of some headers) are written in comma-separated style, easily importable with any spreadsheet editor.

ECO$_2$S raw data are stored in a custom ENE format (extension *.ene*). This is a bundle of two files compressed with a good algorithm (7-zip[4]). The two files in the archive are:

- The ANE file, the actual raw data file, with extension *.ane*.
- The *eco2s_logging.ini* configuration file.

The name of ENE files follows this form:

date_and_time[_id1][_id2].ene

Here, the first part is composed by the date/time of the file, where the minute states the moment acquisition started or ended (depending on the acquisition-time choice) for that file. The second part includes the (mandatory) site ID and the (optional) station ID. The *date_and_time* part of the file name can be in one of the two alternative form:

YYDOYHHMM or YYYYMMDD-HHMM

where the left one is the default. The typical size of an ENE file containing half-hourly EC data of wind, temperature and two scalar concentrations acquired at 20Hz (a total of $\approx$ 36000 data lines) is of about 260 kb.

A.0.1 The ANE data file

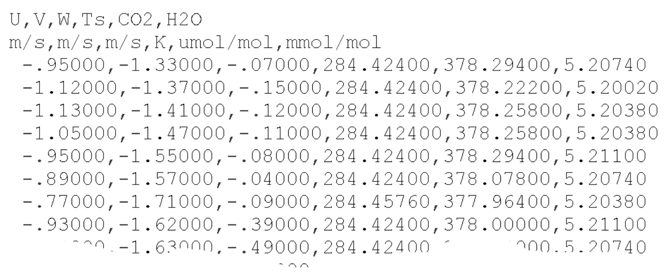
The ANE files contain the actual raw data. The name of ANE files follows exactly the same scheme as the ENE files:

date_and_time[_id1][_id2].ane

with, of course, the exception of the extension. The file itself is a comma-separated data file composed by a 2-lines header where the variables and the units are specified, followed by the raw data columns (see Fig.5).

ANE files contain values in physical units. Anemometric variables are stored in predefined units, regardless of their original units, while eventual

[4]www.7-zip.org. 7-Zip is a file archiver with a high compression ratio. It is an open-source software. Most of the source code is under the GNU-LGPL license.



```
U,V,W,Ts,CO2,H2O
m/s,m/s,m/s,K,umol/mol,mmol/mol
 -.95000,-1.33000,-.07000,284.42400,378.29400,5.20740
 -1.12000,-1.37000,-.15000,284.42400,378.22200,5.20020
 -1.13000,-1.41000,-.12000,284.42400,378.25800,5.20380
 -1.05000,-1.47000,-.11000,284.42400,378.25800,5.20380
 -.95000,-1.55000,-.08000,284.42400,378.29400,5.21100
 -.89000,-1.57000,-.04000,284.42400,378.07800,5.20740
 -.77000,-1.71000,-.09000,284.45760,377.96400,5.20380
 -.93000,-1.62000,-.39000,284.42400,378.00000,5.21100
 [illegible],-1.63000,-.49000,284.42400,[illegible]000,5.20740
```

Figure 5: Example of a few lines from an ANE file

further variables (scalars, temperatures, pressures) are stored with the units specified by the user at acquisition time, within a set of possibilities. All available variables along with their symbols and units are listed in Table(2). Variables are intentionally stored in physical units to provide the user with the possibility to visually inspect the acquired data, quickly without the need for time consuming unit conversions. This brings the disadvantage of not storing the very original information retrieved from the anemometer (and connected sensors). However, this problem is overcome in two ways. Firstly, at acquisition time an option is available for storing the so-called *native streams*, along with the ENE files. Native streams are the very original anemometer outputs, highly compressed and stored on demand in a separate folder (see later, sect.A.0.3). Secondly, all the information used to convert the original anemometric data into the standardised units are contained into the companion file *eco2s_logging.ini* (see sect.A.0.2). These conversion information allow to toggle between physical and native (that can be either physical, electrical or custom) units at any time.

variable	**symbol**	**units**
wind components	u,v,w	m s^{-1}
speed of sound	SoS	m s^{-1}
sonic temperature	Ts	Kelvin
CO_2 mixing ratio or molar fraction	CO_2	μmol mol^{-1}
CO_2 molar density	CO_2	mmol m^{-3}
H_2O mixing ratio or molar fraction	H2O	mmol mol^{-1}
H_2O molar density	H_2O	mmol m^{-3}
air temperature	Te	K or °C
air pressure	Pe	kPa
Sensor cell temperature 1	Ti1	K or °C
Sensor cell temperature 2	Ti2	K or °C
Sensor cell pressure	Pi	kPa

Table 2: Standardised units and variable symbols in the ANE files

A.0.2 The *eco2s_logging.ini* configuration file

The second file contained in the ENE archive is a configuration file compiled by ECO_2Catch at acquisition time. This file contains all the information relevant to the companion ANE file, regarding where and how it has been acquired, the set-up of the station and of the sensors and, in general, all the acquisition-time information related to the raw data. The introduction of the *eco2s_logging.ini* file into the ENE archive, while adding a negligible amount of bytes (it contributes for about 0.1% to the ENE file size), constitutes a major step toward a robust and confident management of raw EC data. In fact, having the *eco2s_logging.ini* file attached to each raw file allows to:

- avoid retrieving information needed for processing the file from any external data source.
- easily store raw-data for future re-processing, even by a person who knows nothing about the site/station set-up at the time the file was acquired.
- to a large extent, simultaneously process files acquired with different set-ups (but possibly referring to the same site/station).

Ideally, the user is not required to consider the *eco2s_logging.ini* file explicitly, as it is created, modified and used silently by all ECO_2S tools. None the less, it is relevant to point out that all information are stored as plain text and retrievable and editable by the user at any time.

A.0.3 The *native streams* data files

Raw eddy covariance data are desirably stored in the ENE format described in sect.A. However, at acquisition time an option is available to store also the very original data output from the anemometer. These files, called *native streams*, are highly compressed with 7zip and stored in the .*native_streams* sub-folder, nested in the main destination directory of acquisition. The redundancy introduced by storing double copies of raw data pays back in terms of robustness of acquisition. In particular, if a mistake was done while setting the ECO_2Catch before acquisition started, ENE files might be affected, and potentially to be discarded. In this case, if *native streams* are available, the processing is still possible, by using ECO_2S as you would do with any other type of custom raw files (see sect.4.2).

B Getting started with ECO$_2$Catch

ECO$_2$S suite is organised in two main programs, featuring independent GUIs, namely ECO$_2$Catch and ECO$_2$S. ECO$_2$Catch allows you to acquire Eddy Covariance raw data while ECO$_2$S is comprised of tools for processing raw data and obtaining a number of desired outputs.

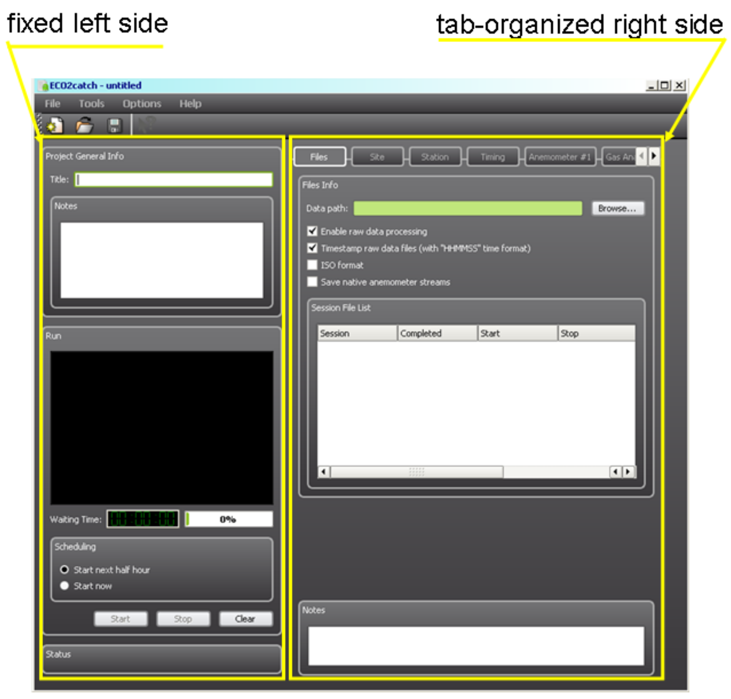
B.1 Downloading and installing ECO$_2$Catch

The latest release of ECO$_2$Catch is available in the ECO$_2$S repository. At the time this document was written, the latest release was ECO$_2$Catch 0.9.8. However, note that the whole suite of software is under continuous development, thus new versions may be released frequently. Installing ECO$_2$Catch is as easy as any other modern program. Locate the downloaded installer, double-click and follow the instructions.

B.2 Running an acquisition session

Double-clicking on the ECO$_2$Catch icon lunches the program. Eddy covariance acquisition with ECO$_2$Catch is organised in *projects*. Ideally, a project contains all the acquisition-time information related to an Eddy Covariance station. A station can feature more than just one EC system, and ECO$_2$Catch projects should allow you to manage all of them. However, the current release of ECO$_2$Catch only supports one EC acquisition per project. None the less, you may run different instances of ECO$_2$Catch to run parallel acquisitions, that will be managed as different projects. All the project-related information are stored in a configuration file in INI format, that you can create, save with a chosen name, re-edit in the future and load to recall your project setup. The configuration file is also editable via a text editor, however we recommend not to do so, as wrong editing may lead to corrupted project files. Before entering any information, you can already save your project via the `File>>Save` menu. We recommend to save all the project files in a dedicated folder, distinct from the raw files destination folder. Although there is no contraindication in doing this, our experience with using the software suggests that having a folder with all the project files saved can be convenient, especially if you run more than one EC tower.

The fixed, left side of the user interface (see Fig. 6) can be used to give a title to the project and to enter project-related notes, such as details of the instrumental configuration, ancillary instruments used at the site and in general all the collateral information that will not be stored in the following. We do suggest to use this box (other ”note” boxes are also available) and to go back to the project notes at the end of the configuration process, to fill here with any critical information you did not have the chance to enter. Notes are not processed by the software, but are stored in a number of



- [✓] Enable raw data processing
- [✓] Timestamp raw data files (with "HHMMSS" time format)
- [ ] ISO format
- [ ] Save native anemometer streams

Figure 6: Snapshot of ECO$_2$Catch user interface

places, such as *each* raw file, for the seek of robustness and data quality assurance in case of future re-processing. On the left side, in the `Run` box you will see the diagnostic output of ECO$_2$Catch during the acquisition. A timer is also provided to support the user in the acquisition process.

The right side of the GUI is organised in tabs, reflecting different conceptual contents and mirroring the hierarchical structure of the underlying INI file. In the following, a quick overlook of the tabs is given. As a general rule, all the yellow fields in the tabs must be entered explicitly before an acquisition session can start, as they are necessary information for further processing and file handling. Thus, as long as not all the yellow fields are entered, the `start` is not enabled.

B.2.1 The `Files` tab

Here you can select the destination folder of acquired files. Actually, this folder will serve as a main folder and a number of sub-folders will be created starting from here, to better organise the huge amount of files that are going to be produced (see later). Afterwards, you can select some options for the acquisition, namely:

- `Enable raw data processing`: if checked, this option forces the software to perform an on-the-fly, *tentative* processing of the acquired files.

This field-level processing can be very useful to check the status of the acquisition (e.g. proper sensors calibration and connections, correct anemometer configuration). However, we strongly discourage from considering these results as definitive, as they are processed with a standard procedure which do not consider the peculiarities of the specific setup. For example, the WPL-term is not included and spectral correction are not applied.

- `Timestamp raw data files`: if checked, a OS-derived timestamp, in the form HHMMSS (*hour-minute-second*), will be stuck at the beginnig of each raw data line.
- `ISO format`: if checked, all file names (ENE files, ANE files and all output files) will be based on the ISO representation of date and time, i.e. YYYYMMDD-HHSS. This is not the default option as, while it is standard, it extends the file name length beyond what is strictly necessary.
- `Save native anemometer streams`: if checked, the very original anemometer outputs are stored, after compression with 7zip, into the sub-folder *.\native_streams*. The typical size of a native stream, for a 30-min acquisition at 20Hz is of about 170 kb. We recommend to always store the native streams, as long as computer space does not limit you. In fact, having the native streams allows you to re-create ENE files with a different *eco2s_logging.ini* configuration file (see sect.A), in case any of the settings entered at acquisition time shows incorrect or inappropriate.

The `Files` tab, and all the other tabs, features a tab-specific note space, that you may conveniently use for entering further details. Note that even if they appear as only one, as they are superimposed, there is a different note box per tab. Finally, the `Files` tab features a box where an overview of acquisition sessions is displayed. An acquisition session starts by pressing the `start` and stops by pressing the `stop` button. The program internally organises the acquired files in such a manner, associates a session ID to each session and displays in the session box the sessions tree, the list of files created with ancillary information such as the starting and ending data/time of the file and a completion indicator (a bullet, green if the file was correctly completed, red if not).

B.2.2 The `Site` tab

Here you feed the program with all the information related to the site under consideration. Beside giving it a full name, you provide an ID, that will be included in the produced files (see sect.A) and enter standard information such as the site coordinates (in WGS84 format, either as sexagesimal or decimal degrees), the altitude a.s.l. and the height of the canopy, that must be greater than zero (enter a very small value, such as 0.01 m, if no vegetation is present at your site).

B.2.3 The `Station` tab

The `Station` tab, conceptually distinct from the `Site` tab as more than one station may coexist at the same site, asks you for details about your installation. Namely, you must enter the height of the anemometer a.g.l. (take the centre of the measuring volume of the anemometer as a reference) and horizontal and vertical separation of (the centre of) the anemometer and the (centre of) gas analyser or its inlet tube, as applicable. Optionally, you can provide a full name for the Station and define a Station-ID. If present, the Station-ID will be added to the Site-ID to form the project-specific (station-specific) file names. Two options are foreseen but not yet available for entering the number of anemometer and/or gas analysers in use. As mentioned at the beginning of this section, so far ECO$_2$Catch only supports acquiring from one anemometer at a time. Multiple acquisitions on the same computer can be ran by running two instances of the program and working with different projects.

B.2.4 The `Timing` tab

Here you select the acquisition frequency and the duration of the raw files. Furthermore you can choose to refer the timing information (of the file names and within the files) to the Coordinated Universal Time (UTC) or to the local, PC-based time. Note that the acquisition frequency you enter here, as well as all the information you will enter in the following tabs, must match the setting of the anemometer in use. That is, setting anemometer configuration in ECO$_2$Catch *is not* setting the anemometer, that instead must be previously configured via the provided firmware or by means of any terminal, using the proper configuration commands provided by the manufacturer in the instrument manual. Presently ECO$_2$Catch does not support direct configuration of the anemometer through the GUI. This is however a desired feature, that is currently under development.

B.2.5 The `Anemometer #1` tab

Here you must enter the anemometer manufacturer/model details (mandatory fields). Optionally you can specify the manufacturer serial number, that can turn out to be very useful when, e.g., a manufacturer publishes information about some fixable bugs in the firmware that applies only to certain serial numbers (as was recently the case with Gill). You can also specify an ID for the anemometer, a sort or internal serial number in your array of traceable anemometers. When you specify the anemometer model, ECO$_2$Catch tries to guess the Serial RS-232 port configuration you may be using. However, you should better check if the serial settings on the right column match your own settings, with special attention to the COM port number, the baud-rate and the flow control method. In the bottom table you must specify all the native anemometric variables output by your anemometer, according to their number, nature and units. Note that here you only enter *anemometric* variables. That is, any eventual analog signal collected by the anemometer and streamed along with anemometric variables in the output strings are not entered here. ECO$_2$Catch supports a subset of the possible anemometric variables, namely the wind components in Cartesian coordinates (u,v,w), the speed of sound, the sonic temperature and the so-called "integer-speed-of-sound", i.e. the actual speed of sound multiplied by 50. While providing a default order, ECO$_2$Catch allows you to choose variables order to match your anemometer configuration. Finally, for each variable you must set the proper units, among those available in the `Units` row. If more than the default four anemometric variables are streamed and must be collected, you can add and edit a column by pressing the + button. `Set Defaults` allows you to save the anemometer configuration and to recall the same settings in the future, when using the same anemometer model.

B.2.6 The `Gas analyser #1` tab

This tab is used to enter relevant information regarding an eventual gas analyser connected to the anemometer. We remind here that currently ECO$_2$Catch only allows acquiring and processing EC data when these are collected in the "anemometer-based" mode. Thus, as a typical situation we expect you to have a gas analyser connected to the anemometer via its ADC channels and that the incoming voltage signals are digitalised by the anemometer, synchronised with the anemometric variables and streamed through the serial port.

Manufacturer and model of the gas analyser are mandatory fields in the `Gas analyser #1` tab. Currently, only Li-cor infrared analysers (IRGAs) are explicitly supported. However for most processing steps, manufacturer-specific features are not critical, thus sensors from other firms can be used, as long as concentration are given in supported units. For firms other the Li-cor, select `unknown` in the manufacturer box. Basing on the IRGA type, the relevant `measure type` must be chosen among molar density, mixing ratio or molar fraction. Optional information are the serial number of the IRGA and a station-specific ID. In case a closed-path IRGA is used, we assume you inlet the sampled air into the cell by means of a tube, for which you must specify the geometry (diameter and length) and the average (or nominal) flow rate. Then, the measurement-specific information must be entered in the bottom table. Similarly to the anemometer table, for each analog signal sent to the anemometer you must enter the nature of the variable (among those available), the voltage units (either V or mV) and the voltage range minimum and maximum. In order to allow a proper conversion into physical units, you must also specify the kind of conversion parameters (either `zero-full scale` or `gain-offset`) that are entered as constants `A` and `B`. Finally, time lag values must be provided to the software, as these information are best guessable at acquisition time. Namely, you must a enter default values for scalar time-lags (with respect to anemometric data), as deduced either by a rough calculation of the transit time of scalars in the inlet tubes (or in the space between the sensors) or by some former experience gained at your station (provided the set up was not changed in the meanwhile). Because default values may not be applicable all the times, you must also enter a time-lag *window*, in the form of a minimum and a maximum time-lag, within which the processing software may find the file-specific optimal time-lag, based on the maximisation of the covariances between the relevant scalar and the vertical velocity. If a maximum of the covariance is not achieved within the time-lag window, the default value is used. As a consequence, different impacts on the calculated fluxes may arise from different window choices:

- choosing a narrowed window around the default value may lead to a frequent use of the default, because the window may be to short for detecting the actual maximising value. This option is thus suitable when you have a strict control or confidence on the stability of the inlet flow, e.g. because you use an active flow controller. Remember, however, that the time lag may strongly depend on the relative humidity of ambient air, regardless of the flow rate. This option makes the calculation faster.

- On the other hand, choosing a broad window, besides making the calculation slower (the difference is however normally negligible), may lead to the use of the actual maximising time-lag, which however may not correspond to the physical concept of aligning data belonging to the same eddies. It may well happen that a spurious maximisation is achieved, that does not reflect a real time coincidence. A trade-off it thus necessary in the definition of the time-lag windows. Note that the use of incorrect time-lags can lead to reductions in the calculated fluxes up to 5-10%!

If more than the default two scalars are to be collected, you can add and edit a column using the + button. Similarly, you can erase a column by selecting it and pressing `-`.

B.2.7 Ready to go!

When all the mandatory fields have been entered, the `Start` button on the left bottom activates, and you can start collecting your EC data. Normally, EC data are collected as 30-minutes files, starting at round times, such as 14:00 or 14:30. This is the default settings of ECO$_2$Catch, meaning that once you press `Start`, the software sets in 'delaying mode', to wait until the next round half-hour. However, when trialling your acquisition, you may want the software to start acquiring immediately, possibly collecting short files (e.g. 1 minute) to check that everything is set properly before you launch the acquisition and leave the station unattended. You can toggle between the two behaviour using the buttons in the `Scheduling` space, right above the `Start` button. Regardless of the chosen scheduling, at the next round half-hour the software will close the current file and start with a new file, to rationally align file timings. There is no option to force the software to ignore this structure, which is intended for increasing robustness of acquisition, limiting the possibilities of human mistakes.

B.2.8 During acquisition

Once started, ECO$_2$Catch provides you several ways to check the status and quality of acquisition. The `Session File List` box shows the list of created files, along with their opening and closing times and a bullet showing if the file is complete (green) or not (red). To have complete raw files, however, does not ensure they contain proper data. To check the quality of acquired data you have several options:

- If you enabled the field-level processing by checking the `Enable raw data processing` button, you can verify the results of the on-line processing by charting the data contained in the *fieldfluxes.txt* file, located in the raw files destination directory. This files contain a number of statistics on the collected data and a first estimation of the turbulent fluxes, see sect. *** for more details.
- If you do not enable the raw level processing, or if you want to give a closer look to your raw data, just unpack the collected ENE files (needs 7zip) and take a look to the data, stored there in physical units.
- If you are in trouble with the two options before, or if you are very confident with your understanding of anemometer raw outputs, you may also look at the *native streams* (sect. A.0.3)

C Development package orientation

This document provides a quick overview of ECO_2S *processing engines*[5] development packages, for programmers who wish to compile ECO_2S projects on their machines and to get an orientation through the source code[6].
As for May 2010, ECO_2S is comprised of a Graphical User Interface (GUI) and several processing engines. The GUI is developed in C++ using the Qt[7] libraries (ver.4.5) and Qt creator (ver.1.3) as the Integrated Development Environment (IDE). The processing engines are developed in Fortran, complying Standard Fortran 2003 and are compiled with gfortran[8] (ver.4.4.0 of 19.12.2008), the open-source, free GNU Fortran compiler. Code::Blocks[9] (ver.8.02) is used here as the IDE.
The current version of ECO_2S is released for Microsoft Windows operating systems. It is known to work properly under Windows XP/Vista and supposed to work also under Windows 7 and former versions, such as Windows 2000/NT. Linux and Mac OS releases can be compiled on demand, although they are not continuously tested, thus proper functioning is not assured. Development is done under Windows XP/Vista.

The Graphical Interface and the processing engines communicate (both directions) by means of plain-text configuration files in INI format. This means that the processing engines can also be ran without the GUI, from the command line, after the relevant configuration files have been properly compiled with any text editor.

C.1 The processing engines development package

The archive named `eco2s_fortran_package_20100510.7z`[10] contains the source code for all ECO_2S processing engines and all that is needed to compile it, in order to obtain the actual executable files. As for May 2010, the available processing engines are:

1. ECO_2S::PreProcessor, for processing raw data;

2. ECO_2S::Tilting, for planar fit calculations;

[5]With *processing engines* we intend executable files called by the graphical interface and used to make any ECO_2S calculation.

[6]WARNING: The size of the source code is rather large, counting more than 50,000 executable lines. Therefore, to be able to orientate in it, one needs to be familiar with the adopted programming languages (Fortran and C++). A minimal knowledge of eddy covariance concepts and processing schemes can also help understand the content-oriented names of routines. Note also that this software suite comes with no design/development quality assurance.

[7]http://qt.nokia.com/products

[8]http://gcc.gnu.org/wiki/GFortran. Gfortran is licensed under the GNU-GPL license.

[9]www.codeblocks.org

[10]To unzip, use 7-zip: www.7-zip.org

3. ECO_2S::SpectralAnalysis, for spectral analysis;
4. ECO_2S::FluxComputation, for computing corrected fluxes;
5. ECO_2S::MeteoFormat, for formatting meteorological files;
6. ECO_2S::TimelagOptimizer, for optimizing time-lag estimations for concentration measured with a closed-path systems[11].

Each processing engine is a different executable file and is therefore organised as a different code development project. The code package is organised in two main folders, containing the actual code and the project files for Code::Blocks development environment.

C.1.1 The source code

The folder `eco2s_src_20100510` contains a set of sub-folders:

1. `src_common` contains source files shared among all development projects;
2. `src_preprocessor` contains source files for ECO_2S::PreProcessor;
3. `src_meteoformat` contains source files for ECO_2S::MeteoFormat;
4. `src_spectralanalysis` contains source files for ECO_2S::SpectralAnalysis;
5. `src_planarfit` contains source files for ECO_2S::Tilting;
6. `src_fluxcomputation` contains source files for ECO_2S::PreProcessor.

Furthermore, the folder contains the following sub-folders:

- `src_timelagoptimizer`, containing the source files for ECO_2S::TimelagOptimizer.
- `src_randomerror`, containing source files for a tool aimed at estimating eddy covariance random errors, according to three different approaches. This tool is presently under development.
- `src_playground`, containing a set of basic source files, creating an ECO_2S-compatible framework for development of new implementations.

Code development projects are process-oriented, organised in a two- (maximum three-) levels hierarchy. A main program file calls operational subroutines (hereafter referred to as L1-subs, or "first level subroutines") in a sequential fashion, following the conceptual steps of data assimilation, correction, elaboration and results production. The main program file is named after the specific engine, e.g. `preprocessor_main.f90` for ECO_2S::PreProcessor.

[11]This tool is currently available only in command line mode and is not supported by the GUI.

Each L1-sub is stored in a different file, that is given the subroutine's name. Names of subroutines recall their conceptual content. If a L1-sub calls level-2 subroutines (L2-subs), these are normally stored in the same file as the L1-sub. However, there are exceptions to this scheme, as some L2- (or L3-) subs are used by several L1-subs. These shared L2/L3-subs are stored in dedicated files. Furthermore, files such as `dir_subs.f90` and `str_subs.f90` contain several low-level subroutines/functions, widely used to handle directories or strings.
In addition, there are modules containing declaration of types and variables shared among all projects. These are: `m_numeric_kinds.f90`, `m_typedef.f90`, `m_logger.f90`, `m_libdate.f90`, `m_dates.f90`, `m_common_global_var.f90`). These modules, stored in the `src_common` folder, must be compiled in the order they are listed here, as they are hierarchically organised. An additional, project-specific module (e.g. `m_pp_global_var.f90` for ECO_2S::PreProcessor) uses them all by using the last one, `m_common_global_var.f90`. Finally, all declared variables are available to the main program by using its project-specific module.

A style note
ECO_2S development packages do not make use of any pre-compiled library. The whole content is carried as source files. However, some modules and subroutines have not been developed by the ECO_2S team, but downloaded from publicly available repositories. Those routines have been not or minimally modified, as they work flawlessly, thus their programming style is different from the rest of ECO_2S code.

C.1.2 The Code::Blocks project files

In principle, ECO_2S package can be compiled in any Fortran developing environment, using any compiler, although using one different than gfortran may imply a few modifications to the code. However, we do not provide any "makefile", rather we strongly suggest to use the free, open-source Code::Blocks IDE for handling ECO_2S engines source code. For this reason, in the folder `eco2s_prj_20100510` we provide the Code::Blocks project (`.cbp`) files[12]. Loading these files in Code::Blocks (possibly with some adjustments to source and destination paths) will provide the programmer with the proper development framework. In particular, the proper compiler options and the correct compiling order will be automatically set by the IDE, sparing the programmer the need to set them manually. If, however, one wants to use his own developing environment, ECO_2S team is available for support during the configuration of the IDE.

[12]project files are not available for the source codes in `src_playground` and `src_randomerror`