

# Python Tool Collection

## Introduction

The purpose of this document is to provide a detailed description of the Python tool collection enclosed in the PROFFAST software compilation. The tool collection consists of several Python scripts guiding the user step-by-step through the evaluation process of the EM27/SUN measurements. For the execution of the scripts, it is required that the programming language Python version 3.7 has been installed correctly with all the modules listed in the header part of each file. Since Python is platform-independent, all the scripts are executable on both a Windows and a Linux OS.

The first script creates the input file needed for the preprocessing and starts the executable (optionally in a parallel mode using several cores). The second script moves or copies the resulting binary files of the preprocessing into the analysis directory. The third script searches, copies or creates the map- and pT-files for each day into the corresponding directory. The fourth script creates the input files based on the provided template files that are used for the pre-calculation of the x-sections and the retrieval software. Finally, the last script merges the resulting output files of the retrieval and creates a new file with a selection of the relevant information of the first file.

The Python scripts, the template files, the input files, and the test files are all prepared in a way that the user can execute them one after the other. The user only has to adapt the path details in the header part of each script.

The last section finally provides a short introduction to the Python scripts for creating GEOMS conform HDF files. The scripts are located into the output directory together with a set of sample HDF files.

# 1 Software Compilation

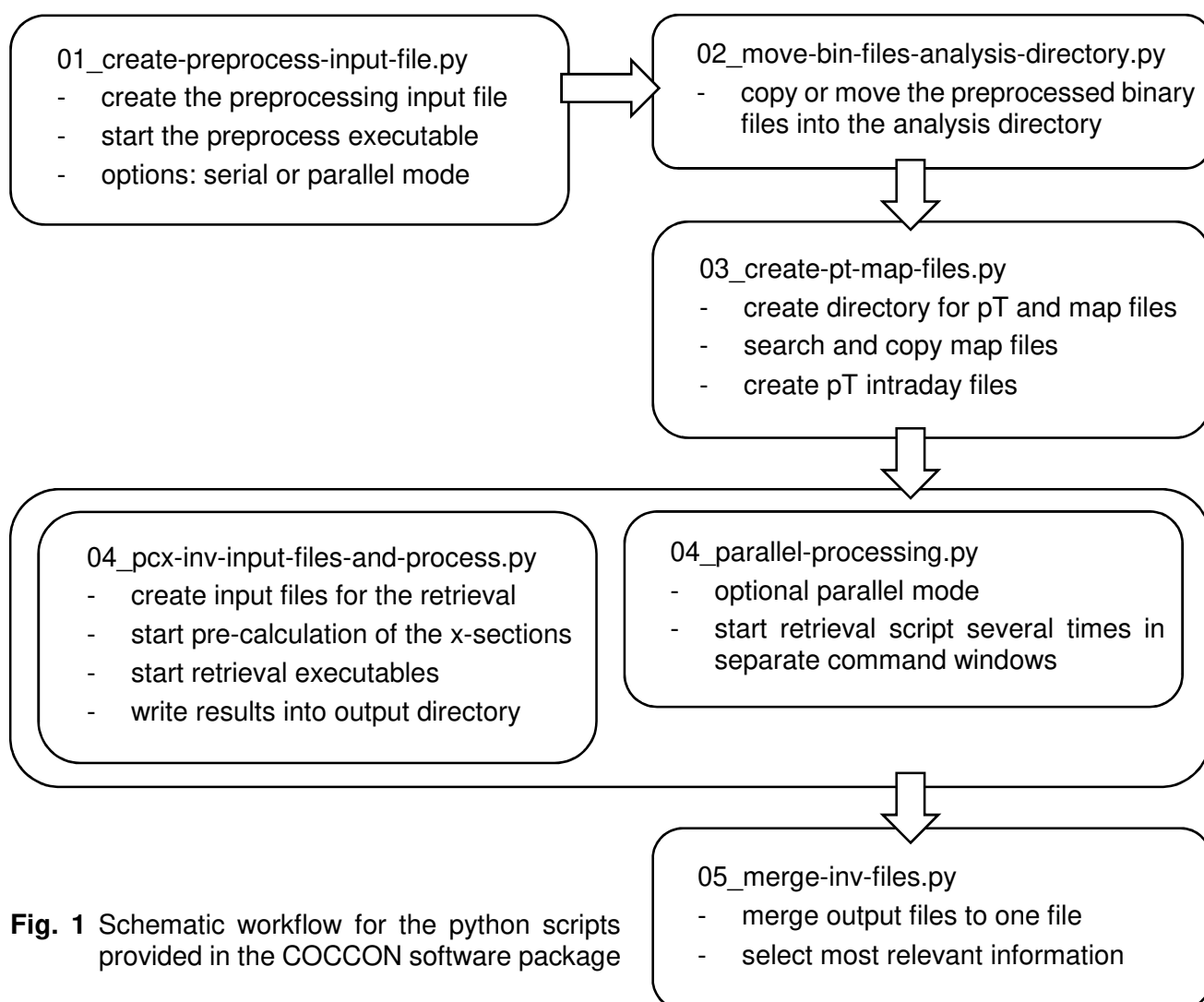
The COCCON software compilation contains the pre-processing executable “preprocess4.exe” as well as the post-processing executables “pcxs10.exe” and “invers10.exe”, some documentation, the source code, a set of test files, and the Python scripts, which are described in this document. The software package can be downloaded directly from the COCCON homepage (see <https://www.imk-asf.kit.edu/english/COCCON.php>) following the link “Data Processing” at the sidebar. After the zip-file has been extracted, the executable files can be run without installation. The file structure of the package is as follows:

analysis	This folder contains the pre-processed binary files (i.e. *.bin), the map-files and the pressure input files.
auxil	This directory contains the template files for the input files “preprocess4.inp”, “pT_intraday.inp”, “pcxs10.inp”, and “invers10.inp”
docs	Some documentation is stored in this folder.
inp_fast	The input fast folder is used for the input files “pcxs10.inp”, “invers10.inp”, and “pcxslev.inp”.
inp_fwd	The spectroscopy (i.e. the HITRAN files), the masses of the molecules, and a temperature parameterization are located in the input forward directory.
out_fast	The retrieval results are saved in the output fast folder.
preprocess	The preprocess directory contains the executable “preprocess4.exe”, the input file for the preprocessing, the “diagnosis” subfolder, and the raw OPUS files.
source	The source code of the pre-processing and the post-processing are in the “source” directory.
wrk_fast	Interim results are saved in the work fast folder.
	The main folder contains the post-processing executables “pcxs10.exe” and “invers10.exe” as well as the Python scripts.

## 2 Python Tool Collection

The PROFFAST software compilation provides a user-friendly tool collection of Python scripts guiding the user through the entire evaluation process of EM27/SUN measurements. Moreover, these scripts support the processing of larger amounts of measured data optionally in a time saving parallel mode. The application of the tool collection requires a correctly installed Python version 3.7 including several modules listed in the header part of each file. Besides the standard modules, the following modules have to be included additionally: os, sys, glob, math, shutil, pandas (up to version 1.0.1<sup>1</sup>), subprocess, time, datetime, and jdcal. The programming language Python is platform-independent. Therefore, all the scripts are applicable on a Windows OS as well as on a Linux OS.

The distributed PROFFAST software package already contains the recommended file structure. The measured pressure and temperature files recorded by the data logger are stored in the “log-files” subdirectory, the a-priori profiles of the trace gases in the “map-files” subdirectory, and the templates for the input files in the “templates” subdirectory. All these subdirectories are included in the “auxil” directory. The Python tools are located in the main folder “prf96-EM27-fast”. The files can be executed within the command prompt using the Python command followed by the file name, i.e. “python file.py”. The sections of this chapter contain the description of each Python script or rather of each step of the data analysis.



**Fig. 1** Schematic workflow for the python scripts provided in the COCCON software package

<sup>1</sup> As pointed out by Paolo Castracane (RHEA Group, ESA/ESRIN, Italy), only the pandas version up to 1.0.1 can be used for the provided Python scripts, while the latest versions will produce an error message

## 2.1 Preprocess4

The first Python script labeled “01\_create-preprocess-input-file.py” creates the input file for the preprocessing at the beginning and starts the executable optionally in a parallel mode subsequently.

The preprocess template file “template\_preprocess4.inp” in the “template” directory is used to get the header part of the input file, i.e. the file “preprocess4.inp”. The header part contains certain criteria for the DC value of the signal, the ILS parameters (i.e. the apodization and phase error) of the first and optionally second channel, the site name, the time offset, the observer coordinates and a section for the comments. All these parameters have to be adapted to the instrument and the location of the site before starting the script.

The second (or last) section of the input file is a list of paths to the OPUS files of the data set to be processed. The global path to the data set of EM27/SUN measurements needs to be adapted in the header part of the script. The OPUS files have to be stored in a separate subfolder within the “preprocess” directory. All the files of a certain measurement day are put together again in separate subfolders labeled with the date using the format “yymmdd”.

After executing the script, all measurements within this global path are appended to the file list automatically. The input file is terminated by the character sequence “\*\*\*” and saved into the “preprocess” directory. This directory also contains the executable file for the preprocessing. Finally, the executable “preprocess4.exe” (for Windows) is started within the same command prompt either in the normal (or sequential) mode processing the files one after the other or optionally in a parallel mode with several command windows.

### 2.1.1 Preprocess Input File

After executing the Python script and creating the preprocessing input file, the user is asked whether he wants to run the preprocess software or not. If so, he will then be asked whether he wants to start the program in a parallel mode or not. If the user decides against it, the program is executed in a normal mode, which means that the files listed in the input file are processed sequentially using the same command prompt. The resulting binary files of the preprocessing are stored in a new subfolder “cal” using the local time of the measurement with the format “hhmmss”, an additional character for the first channel “N” or if available for the second channel “M”, and the file extension \*.bin. Please note that this time is not exactly the same time stored in the OPUS files, which is the beginning of the measurement, but the mean time of the start and end times (i.e. adding 29 seconds to the start time).

### 2.1.2 Parallel Processing

In case of processing the data in the parallel mode, specification of the desired number of parallel jobs is required. The maximum number of jobs is initially limited to eight. For optimally matching the CPU architecture, the number can be raised to higher values in the last section of the Python script. The preprocessing executable is started with two parameters, i.e. “preprocess4.exe jobs jobnr”. The first parameter is the total number of jobs and the second parameter the job number used for this sub-processing, which means that only a certain part of the files listed in the input files is actually processed. This means basically, that the whole file list can be subdivided into several parts and processed parallel in different command windows at the same time without interrupting each other.

Due to the quality checks and unexpected errors, a warning message may occur expecting to enter “0” to shut down the program or “1” to go on. In order not to interrupt the processing each time this message appears, one can read in the file “continue.txt” which contains only the character “1” in each line. The alternative command is then “preprocess4.exe < continue.txt”. Both measures can reduce the processing time significantly (approximately by the number of jobs).

## 2.2 Copy or Move Files

The second Python script “02\_move-bin-files-analysis-directory.py” moves the pre-processed binary files performed in the first step to the “analysis” directory of the main PROFFAST folder. As before, the source and destination path in the header part of the Python script has to be changed. Optionally, the code line for moving the files can be commented out, and replaced by the line that for copying the files instead. Finally, a new folder labeled “pT” is created next to the existing “cal” directory for each day folder. These empty directories are required in the next-following step.

## 2.3 PT- and Map-Files

The third Python script labeled “03\_create-pt-map-files.py” basically consists of two parts. At first, a directory scan for the map files of each measurement day is performed. The location of the map-files has to be specified in the header part of the Python script and is usually expected to be in the subdirectory “map-files” of the “auxil” directory. In case of an existing map file, a copy into the corresponding “pT”-subfolder of the day folder is created. The map files contain the a-priori profiles of a certain site and date for all trace gases that are of importance for the retrieval process.

The map files are created in the post processing after running the GFIT retrieval software. Alternatively, these files can also be generated on the centralized mod maker server provided by the TCCON community. For further information on this issue please visit the TCCON homepage ([tcon.caltech.edu](http://tcon.caltech.edu) or <https://tconwiki.caltech.edu>).

Secondly, this Python script searches for the data logger files containing the local pressure and temperature values for each measurement day. The location of the pT-files is expected to be in the “log-files” subdirectory of the “auxil” folder. Again, this path has to be adjusted in the header part of the script. Once, the pT-file is found, a file named “pT\_intraday.inp” is created and saved into the corresponding “pT”-subfolder. This file is composed of the “template\_pT\_intraday.inp” template located in the “template” directory and the content of the data logger file.

## 2.4 Pcx10 and Invers10 Execution

Based on the preparatory work of the previous scripts, the retrieval software can now be started either in a normal (sequential) mode or in a parallel mode. The forth script “04\_pcx-inv-input-files-and-process.py” can be used to process the data in the normal mode. All the pre-processed files of the subdirectory in the “analysis” folder are then evaluated sequentially. In the case of parallel processing, the processing time can be reduced significantly, as already pointed out in the context of the parallel pre-processing of the raw data. A list of files to be processed is subdivided into a certain number of jobs. These jobs are processed in different command windows at the same time. Here, the Python script “04\_pcx-inf-parallel-processing.py” has to be used instead. This script starts the “04\_pcx-inv-input-files-and-process.py” scripts in each of the new command windows. The number of parallel jobs is initially limited to eight and can be extended according to the needs of the user. In both Python files, the path information has to be adapted before executing the scripts.

### 2.4.1 Pcx10 and Invers10 Input Files

The script for the normal mode “04\_pcx-inv-input-files-and-process.py” starts at first the executable “pcxs10.exe” for the pre-calculation of the x-sections for each day. Beforehand, the file “template\_pcx10.inp” in the template directory is used to create the requested input file “pcxs10.inp”. The resulting input file is saved in the “inp\_fast” directory for each day separately. The template file contains several sections with all the requested information for

the calculation of the x-sections. The most relevant sections for the user, which have to be adapted, are the details on the observer location in the first section, the path to the map file in the second section, and the paths to the map files in the second last section for the volume mixing ratios (VMR). Once, this task is completed for a certain day, all the measurements of this day can be evaluated in the second step.

The execution of the “invers10.exe” program requires as before an input file, which is created out of the template file “template\_invers10.inp” within the “template” directory. This template file does not need to be edited or changed in general. It contains a list of files at the end, which is generated for all pre-processed data of a certain day automatically. Again, the resulting input file is saved in the “inp\_fast” directory for each day. During the post-processing, the results are stored in the “out\_fast” directory using a file appendix \*-colsens.dat, \*-invparms.dat, and \*-job01 to \*-job05. The vast majority of users only require the contents of the \*-invparms.dat” files. Here, the Julian date, the ground pressure and temperature, the apparent solar zenith angle, the retrieved column-averaged dry air mole fractions (i.e. XAIR, XCO<sub>2</sub>, XCH<sub>4</sub>, XH<sub>2</sub>O, XCO), and the total columns of all trace gases of relevance (i.e. H<sub>2</sub>O, HDO, CO<sub>2</sub>, CH<sub>4</sub>, N<sub>2</sub>O, CO, HF) are provided for each measurement day separately. Since the results are encrypted with names like “gas” (gas01 to gas08) or “job” (job01 to job05), they are translated and summarized in the last step (see last section 2.5). This step also includes a merging of all measurement days for better evaluation afterwards.

#### 2.4.2 Parallel Procession

For the execution of the retrieval in a parallel mode, a further Python script labeled “04\_parallel-processing.py” is provided. Once again, the paths in the header part of the script have to be adjusted before starting the script. At first, the user is asked whether he wants to start the retrieval in parallel mode. If so, he will then be asked how many jobs should be executed at the same time. The maximum number of jobs is restricted to eight. Nevertheless, this number can be raised within the Python script as required by the user. Depending on the number of jobs, several command windows pop up and close again after the processing has been finished.

## 2.5 Merge Output and Select Content

The last Python script “05\_merge-inv-files.py” is basically composed of two parts. In the first part, all the output files are simply merged to one file without any changes in the content. The name of this file is the core name of all output files (e.g. sod2017\_em27sn039.dat). In the second part, all the relevant information of the output files (i.e. the \*-invparms.dat files) are selected, whereas the total columns labeled as “job0X\_gas0Y” are renamed (see table below). Finally, the results of all measurement days are merged into one file with the name extension “selc” (e.g. sod2017\_em27sn039\_selc.dat).

The column selection includes the Julian date, the date/time, the ground pressure and temperature, the apparent solar zenith angle, the column-averaged dry air mole fractions XAIR, XCO<sub>2</sub>, XH<sub>2</sub>O, XCH<sub>4</sub>, and XCO, and the renamed total columns of the trace gases H<sub>2</sub>O, HDO, CO<sub>2</sub>, CH<sub>4</sub>, N<sub>2</sub>O, CO, O<sub>2</sub>, and HF. The renaming follows the scheme:

job01_gas01	H2O	job04_gas04	CH4
job02_gas07	O2	job05_gas06	CO
job03_gas03	CO2	job05_gas04	CH4_S5P



### 3 Practical User Guidance for HDF Files

This section provides the user with additional information about the Python version, the required modules and finally the practical application of the Python scripts for creating GEOMS conform HDF files. Initially, a current version of the PROFFAST software package is required, which provides all necessary test files for creating HDF files. The input data for the scripts are located in the analysis folder (i.e. the “pT\_fast\_out.dat” and “VMR\_fast\_out.dat” files in the “pT” directory) and the output folder (i.e. the “\*-invparms.dat” and “\*-colsens.dat” files in the “out\_fast” directory).

For all previously introduced scripts, a Python version 2.7 is required. It is important to note, that the latest Python version (i.e. version 3.x) will most likely not work. An adjustment of the scripts to the latest Python version may be considered at a later stage. In addition, numerous modules are included in the header of the scripts. These modules have to be installed before executing the scripts. The required modules are:

- os, time, shutil, glob, math, re, collections
- h5py (recommended version 2.10.0)
- numpy (recommended version 1.16.5)
- nested-dict (recommended version 1.61)
- datetime (recommended version 4.3)
- pyhdf (recommended version 0.10.2) , pyhdf.SD

The Python scripts for creating HDF files are located in the output directory (i.e. “out\_fast”):

- CocconCSVtoGEOMS.py
- CocconDef.py
- CocconClass.py
- CocconCSVtoHDF.py
- CocconHDFtoGEOMS.py

For executing the scripts, the user only has to adjust the code lines for the location of the analysis and the output directory (see line 41 and 42 in the script CocconCSVtoGEOMS.py):

- ana\_path = r'D:\prf96-EM27-fast\analysis\sod2017\_em27sn039'
- out\_path = r'D:\prf96-EM27-fast\out\_fast\sod2017\_em27sn039'

With the correct settings, the execution of the scripts on a command prompt is realized by the command line: “python27 CocconCSVtoGEOMS.py”. For each measurement day, the directories are scanned for the required files for generating the HDF files. The file list is then processed successively. The HDF files are stored in the same output directory with a subdirectory labeled as “sodankyla/EM27SUN”. For opening or editing the HDF files one can use e.g. the software tool “HDFView”.

For new sites further adjustments have to be made, i.e. the site name in the script “CocconCSVtoGEOMS.py” (see line 32), the site coordinates in the script “CocconCSVtoHDF.py” (see line 33), and the names, affiliations, addresses, and emails of the principal investigators (PIs), the data originators (DOs), and the data submitters (DSs) in the script “CocconHDFtoGEOMS.py” (see line 131).