**PAPE-P**oint **A**nd **P**uff Concentration M**E**asurements Analysis Program

User’s Guide

**Summary:**

This program, written in python, analyses and creates plots and statistical descriptions of boundary layer wind tunnel concentration measurements. The program differentiates between two types of concentration measurements, namely between a continuous release measurement (where the tracer gas is released continuously over time) and a puff release measurement (where the tracer gas is released in many regularly repeated short intervals, called puffs).

1. **Requirements**

**1.1 Dependencies**In order to function correctly, PAPE requires python 3.8.2, with the following packages additional installed:

numpy v1.18.3

scipy v1.4.1

pandas v1.0.3

logging v0.5.1.2

matplotlib v3.2.1

skimage v0.16.2

openpyxl v3.0.3

Newer versions of python and the required packages should not impact the functionality of the program, please contact the developers if you encounter any issues with versions of python or the above packages newer than the ones listed above. However, backwards compatibility (i.e. compatibility with older versions), especially with python 2.x or older, cannot be guaranteed!

The program was tested using Spyder 3.3.6 on a Computer running Ubtuntu 20.04LTS, but should work on any distribution that supports the above. Note that the purpose of the GUI installation (see Section 2.1) is to run the program without the use of Spyder.

**1.2** **Installation**

To install PAPE in your computer, copy the program files to a directory of your choice. It is recommended to install the program in a directory to which you have read and write permission to avoid potential permission conflicts. If you want to run PAPE via the GUI, make sure that your system is set to allow the execution of python files as a program. The procedure to adjust this varies a bit dependeing on your OS, but a first step is always to right click on the file titled “PAPE\_GUI.py” in the PAPE root directory, and make sure that the option “Allow executing the file as program.” A second step would be to go to (on Linux) “Preferences → Behavior” in your file explorer and under “Executable Text Files,” select either “Run them” or “Ask what to do.” The exact steps may vary a bit on other operating systems, but the principles settinsg should be the same.

After following these steps with the required dependencies in section 1.1, PAPE should be installed and running on your computer.

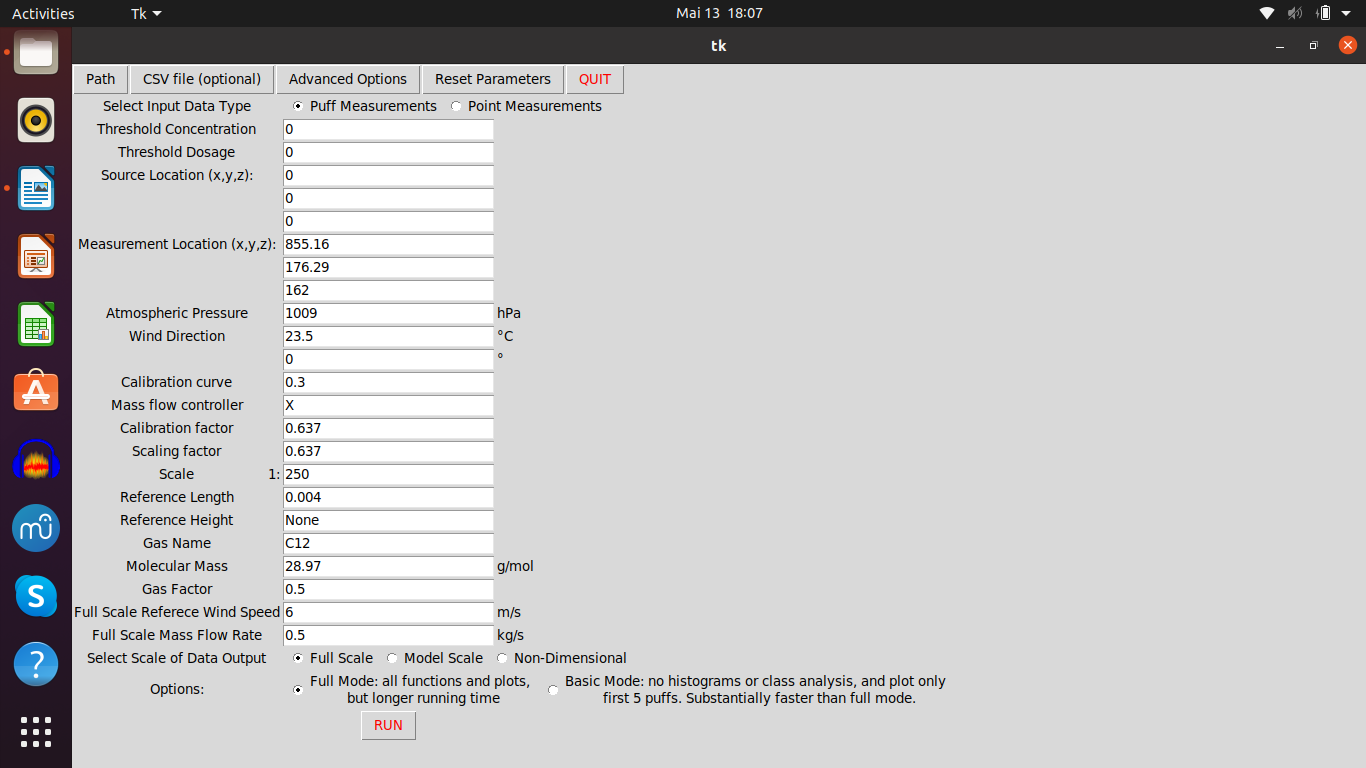
1. **Getting Started**

The program can easily be operated and started by running one of several source code scripts, but also features a graphical user interface (GUI) for easier operation of the program. Note that most of the development occurs primarily on the source code; thus, not all current features may be available when operating the program through the GUI. Note that it is substantially easier to operate the program incorrectly through the source code than through the GUI. For this reason, **it is reccomanded that users with little or no experience with python programming use the GUI version the program.**

**2.1: Graphical User Interface (GUI)**

To run the program using the GUI, follow the steps above to allow for the exection of pyethon scripts as exectuables, then double click on the file “PAPE\_GUI.py” in the PAPE root folder. If a dialog appears asking you what to do with the script, select “Run” (exact wording may vary slighly depending on the OS). If you want to see what the program is doing and what it outputs you can also select “Run in Terminal.” This option is most likely to be suitable only for experienced users, but has much less potential to do damage to the actual program than running and editing the program from a CLI.

Upon opening the program, you should see a window pop up which looks very similar to Figure 2.1. In the main window towards the left are the program options and a priori values. **The “default” values” which are loaded upon starting the program come from an arbitrary dataset and are by no means**

Figure 2.1: Initial window of the PAPE GUI. ****

**universally applicable!** They are just placeholders to make sure that there is “something” in these values, and should be edited to match the dataset being analyzed. A detailed explanation of the physical meaning of the options and a priori values is provided in Sections 3 and 4. Note that the two option “Select intpu data type” correspond to switching between the example\_point\_concentration.py and example\_puff\_concentration.py scripts (see Sections 2.2 and 3).

To run the program, you must first select one or multiple files for the program to analyze. In GUI mode this is accomplished by clicking on the “path” button at the top left corner. After selecting the data location, a selectable list of files which the program recognizes for analysis pops up on the right hand side of the main window. **If none of the files are selected, the program does nothing but stays open.** Currently, only files with the extension .txt.ts#0 are supported. If you would like support for more data please contact the developers.

PAPE also supports specifying the a priori information through a csv file; this option is recommended for larger datasets with multiple files. You can specify the location of the file containing the a priori information through the “CSV file (optional)” button next in the top left of the window. The csv file has several formatting requirements, so a sample csv file containing the a priori information is located in the /windtunnel subfolder of your installation directory (currently in the root directory of PAPE). The sample file is called Q2\_Ambient\_Conditions.csv.

The a priori values can be reset to their default values by clicking on the “Reset Parameters” button at the top right of the window.

PAPE also has two options for regulating the scope of the data analysis performed.

-”full mode,” which runs the entire set of analysis methods, including ensemble and class analayis. This mode also outputs a full set of plots, including a separate plot for the concentration time series during each individual puffs. **Running PAPE in “full mode” can be very time- consuming, especially for large datasets.**

-”basic mode,” which leaves out the class analysis, and consequently the histograms, entirely. This mode also plots only the first five puffs. **Running PAPE in “basic mode” is substantially faster than “full mode,” and is recommended for most purposes.**

These options can be set directly from the GUI. PAPEs default setting is to run in basic mode.

The “Advanced Options” button opens a new subwindow which pulls up a range of additional settings for the data analysis. **Do not modify these settings unless you know what you are doing.** Incorrectly modifying these parameters can several compromise the integrity of the output of this program. If you want to dive into these more advanced parameters, it is highly recommended to get started reading the description of the variables time\_threshold and n\_outliers in section 4.

**2.2 Command-Line Interface (CLI)**

To run PAPE in the command-line interface mode (CLI), start by opening the file example\_point\_measurement.py (or example\_puff\_measurement.py for puff measurement, if you wish), either in spyder or using a text editor. Enter the path of the dataset (variable ‘path’) and filename(s) (variable ‘namelist’) of the dataset you wish to analyze. You should also specify the a priori information located below the namelist variable. A detailed explanation of the physical meaning of the options and a priori values is provided in Sections 3 and 4. While the program will in theory run without any a priori values, correct a priori information is required for physically correct results. Of particular importance are the variables x\_source, y\_source, z\_source, x\_measure, y\_measure, z\_measure, pressure, temperature, scaling factor, scale, ref\_length, and ref\_height. Furthermore, you should specify the outputs scale of the data, by setting the variable full\_scale to either “fs” (full-scale output), “ms” (model scale output), or “nd” (non-dimensional output). Now you can run whichever file you just edited (in spyder just click ‘run’ in the top toolbar), and the program should output the results of your analysis.

Note that if you want to analyze more than one dataset (i.e. if you have more than one entry in namelist), you may want to consider inputting the a priori information into a csv file. The csv file has several formatting requirements, so a sample csv file containing the a priori information is located in the /windtunnel subfolder of your installation directory (currently //ewtl2/work/Johannes/Konzentrationen/windtunnel). The sample file is called S04\_Ambient\_Conditions.csv; if you rename this file you must also enter the new name into the python script (variable csv\_file, line 18 in the point measurement script and line 21 in the puff measurement script). In the csv file, the first line must contain the filename of each dataset, and this must be identical to the filenames of your input datasets (see sample file). If this is not the case, the program will resort to using the a priori values written in the respective python script. Note further that you must enter (proper) values for all variables located in the sample csv file, you cannot selectively specify some variables from the csv file and others directly from the python file. If the csv file does not contain values for all variables located in the sample csv file, the program will once again resort to using the values from the python script for all the a priori values (thus ignoring the values in the csv file). Everything else is taken care of by the program.

PAPE also has two options for regulating the scope of the data analysis performed. These are controlled by the variable “functions\_mode”

-”full mode,” (functions\_mode=”full”) which runs the entire set of analysis methods, including. ensemble and class analayis. This mode also outputs a full set of plots, including a separate plot for the concentration time series during each individual puffs. **Running PAPE in “full mode” can be very time-consuming, especially for large datasets.**

-”basic mode,” (functions\_mode=”basic”) which leaves out the class analysis, and consequently the histograms, entirely. This mode also plots only the first five puffs. **Running PAPE in “basic mode” is substantially faster than “full mode,” and is recommended for most purposes.**

Note that the output is, per default saved in the same location as your original dataset (i.e. under the path you entered for the variable ‘path’). While the point measurement script outputs mainly text files, the puff measurement script outputs several (i.e. many) plots in several subfolder. All necessary subfolders are automatically generated by the program.

1. **Puff and Point Concentration**

A key element of this data analysis package is the ability to analyse both continuous release (point concentration) and puff release (puff concentration) timeseries separately. It is up to the user to determine which type of tracer gas release was used for a particular time series; it is also critical to determine this **before** running this program. The two analysis modes cannot be used interchangeably under any circumstances! **If the incorrect analysis mode is selected, the program will appear to run and output normally, but the output will make as much sense as replacing the 16 anvils in Richard Wagner’s *Das Rheingold* with 25 Hautboys!** As stated above, the continuous release mode implies that the tracer gas was released over the entire timeseries at a steady, continuous rate into the wind tunnel. This type of release is particularly useful when studying, for example, the scenario where harmful chemicals (usually gases) are released from an industrial plant as a result of a malfunction (i.e. leakage). Analyzing this type of data is comparatively straightforward and the program will usually run very quickly (i.e. not more than a few seconds) unless a large amount of input data is specified (>100 mb).

The puff release mode is quite a bit more complicated, but in many cases also more realistic for many applications. Here, instead of being released continuously over time, the tracer gas is released only during short (usually lasting not more than a few seconds), but repeated intervals. Each of these intervals is called a puff. For the purpose of diagnosing the characteristics of the individual puffs, the program plots, among other things, the release of the tracer gas over time into the puff plots. Note that the release signal is binary; i.e at a point in time the gas can either be released or not released. The mean release signal, however (plotted in the plot of the mean puff) can give crucial information about the release length in each puff; if the release lengths for the individual puffs are all the same, the mean release signal appears will appear as a rectangle function (as do all the individual release signals for a particular puff). If the puff release lengths are all different, the mean release function will have a non-rectangular shape. Note that due to the binary nature of the release signal equal release lengths are also necessarily implied if the mean release signal appears as a rectangle function (this could not be generally said if the release signal could take on multiple, in particular a continuous set, of values). The mean puff for an example time series is shown in figure 1.

1. **A priori and input values**

This section briefly describes, albeit in some more detail than in the above section, the input and a priori variables. The following a priori variables are specified in both the puff and point concentration analysis:

x\_source: x-position of the emission source, in mm (model scale). Note that this is **not** necessarily the same as the measurement location (see below).

y\_source: y-position of the emission source, in mm (model scale). Note that this is **not** necessarily the same as the measurement location (see below).

z\_source: z-position of the emission source, in mm (model scale). Note that this is **not** necessarily the same as the measurement location. While PAPE will allow negative z-coordinates, **any negative values for z\_source should be approached with extreme skepticism, as any negative value for z\_source implies that the emission source is below the wind tunnel surface.**

x\_measure: x-position of the measurement location, in mm (model scale). Note that this is **not** necessarily the same as the source location (see above).

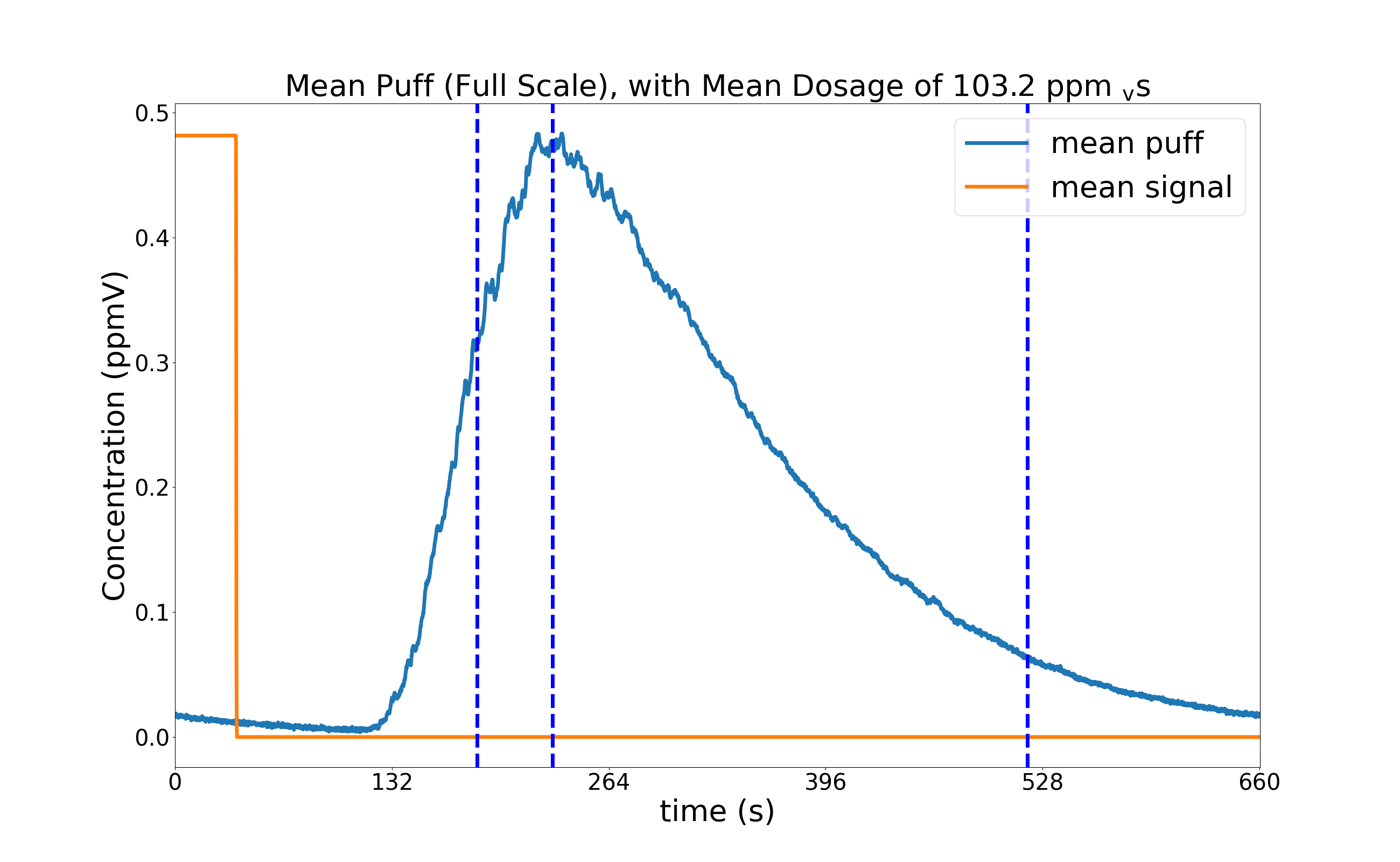


Figure 1: Mean puff for example time series. The puff signal is shown in blue (units ppmV) and the release signal in orange (binary, units adjusted to make signal readable on plot).

y\_measure: y-position of the measurement location, in mm (model scale). Note that this is **not** necessarily the same as the emission source (see above)

z\_measure: z-position of the measurement location, in mm (model scale). Note that this is **not** necessarily the same as the the source location (see above). While PAPE will in theory allow negative z-coordinates, **any negative values for z\_measure should be approached with extreme skepticism, as any negative value for z\_measure implies that measurements are taken below the wind tunnel surface.**

pressure: reference air pressure, **outside wind tunnel**, during measurement, in Pa

temperature, reference air temperature**, outside wind tunnel**, during measurement, in °C.

calibration curve: tbd

mass flow controller: tbd

calibration factor: tbd

scale: scale of model. Note that the number here denotes the real-world units represented by one corresponding model unit; i.e. a scale of 1:250 would equal a value of 250 for the parameter ‘scale.’ Note that this is not the same as the output scale controlling variable “full\_scale.”

ref\_length: reference length, in m (model scale), used to make data non-dimensional

ref\_height: reference height, in m (model scale), use (in theory) for constructing wind profiles. Currently not used.

gas\_name: Name of tracer gas. Currently only used for documentation purposes.

mol\_weight: molecular weight, in kg/mol, of gas. Units here are highly doubtful, **to be verified**. More likely units are g/mol.

gas\_factor: used to calculate model scale flow rate

full\_scale\_wtref: full scale reference wind velocity, in m/s, used to calculate full\_scale concentration and time.

full\_scale\_flow\_rate: model scale flow rate out of emission source, in kg/s. Note that this is later converted to full scale flow rate, in m3/s. Convention of naming both the full scale and model scale flow rate full\_scale\_flow\_rate taken from Benyamin Schliffke’s original script, the reason for this rather odd naming convention is at this point unknown.

The puff analysis uses all of the a priori variables above, plus the following additional a priori variables:

threshold\_concentration: the minimum peak concentration, in ppmv(model scale) of a puff to be included in the statistical analysis. All puffs with a peak concentration below this value will be filtered out of the analysis.

threshold\_dosage: the minimum dosage, in ppmv(model scale), of a puff to be included in the statistical analysis. All puffs with a peak concentration below this value will be filtered out of the analysis.

n\_exclude: number of outliers to exclude. The default setting, None (without quotation marks of any kind), automatically selects the number of outliers to remove. It is explicitly noted that setting this variable to None does not mean that no outliers are removed from the datasets. Furthermore, changing **this variable can substantially compromise the integrity of the results of the program. Do not change the value of this variable unless you know what you are doing! In all cases, be sure to properly document and justify any changes to this variable when applying the results of this program to any work.**

full\_scale: variable which controls which scale to perform the analysis in. Accepts ‘ms’ (for model scale data), ‘fs’ (for full scale data), and ‘nd’ (for non-dimensional data) as inputs.

functions\_mode: determines the scope of the analysis. Accepts ‘full’ for complete, usually time-consuming, analysis, and ‘basic’ for a trimmed-down, but usually faster, analysis. See Section 2.2 for more details.

time\_threshold:a number which determines at which percentages of the total dosage the arrival and leaving times should be computed. The number itself is the fraction of the dosage at which to define the arrival time as, with the leaving time determined in a symmetric manner. For example, setting this variable to 0.01 will define the arrival time at 1% of the dosage, and the leaving time at 99% (or 100%-1%) of the dosage. The agreed-upon setting for this variable is 5%, but some puffs cannot be appropriately characterized by this convention. **Warning: changing this variable will substantially modify the results of the program, and can severely compromise the integrity of the resulting analysis. Do not change this variable unless you know what you are doing!! In all cases, be sure to properly document and justify any change to this variable in any work that results from the usage of this program.**

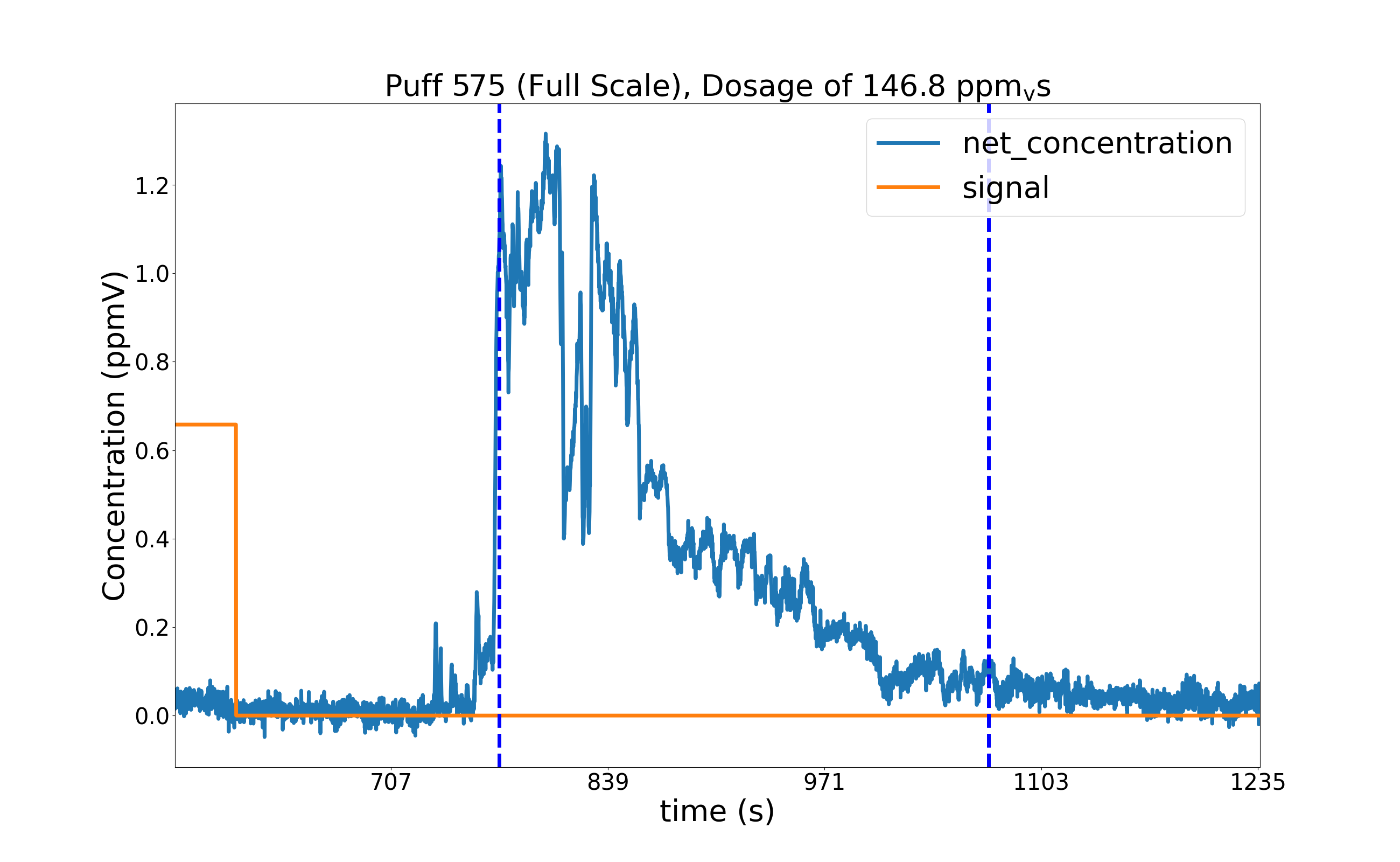
1. **Output**

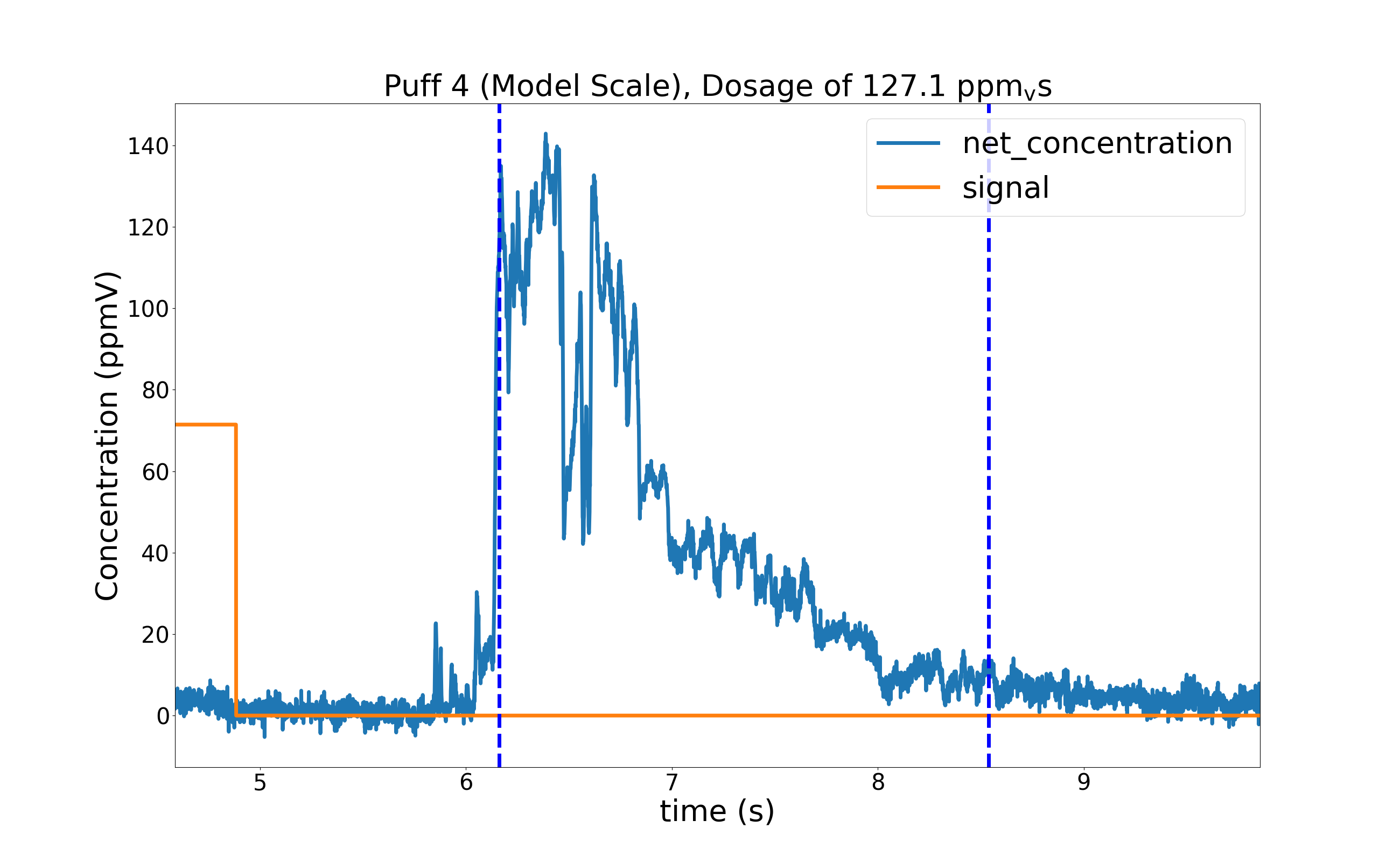
*5.1 Point Concentration*

In point concentration mode (i.e. continuous release) the program outputs three txt files containing the results of the analysis. The files begin with \_avg\_ (average data, in full scale), \_fs\_ (time series of full scale data), and \_ms\_ (model scale data). Further, the program outputs two plots of the frequency distributions (both relative and cumulative) of the net concentration values. Both the txt files and the plots are saved in the original path of the data (i.e. whatever is specified under the a priori variable ‘path’).

*5.2 Puff Concentration*

In puff concentration mode (i.e. puff release) the output of the analysis is substantially more extensive then in point concentration mode. All data is saved in a subfolder titled Puff\_Plots/data\_series under the original path, where data\_series is the name of the data series as specified in the a priori variable name. Note that a separate folder is generated for each time series; this is done to avoid having too many plots in one subfolder. First and foremost a separate plot is plotted for each individual puff, which plots the time series of the net concentration signal, as well as the release signal (see Figure 1) and dosage (in the plot title). The plots are labeled by the start time of the puff release at each plot, for this reason, a single puff may have different numberings in full scale, mode scale, and non-dimensionally. The arrival time and leaving time are plotted by the two dashed lines (note that the leaving time is always necessarily greater than, or in a few rare cases, equal to, the arrival time). Figure 2 shows a plot from a puff from a sample time series, in full scale, model scale, and non-dimensionally. Note that time is always the same as specified in the time series, and not the start of the individual puffs! Note that the





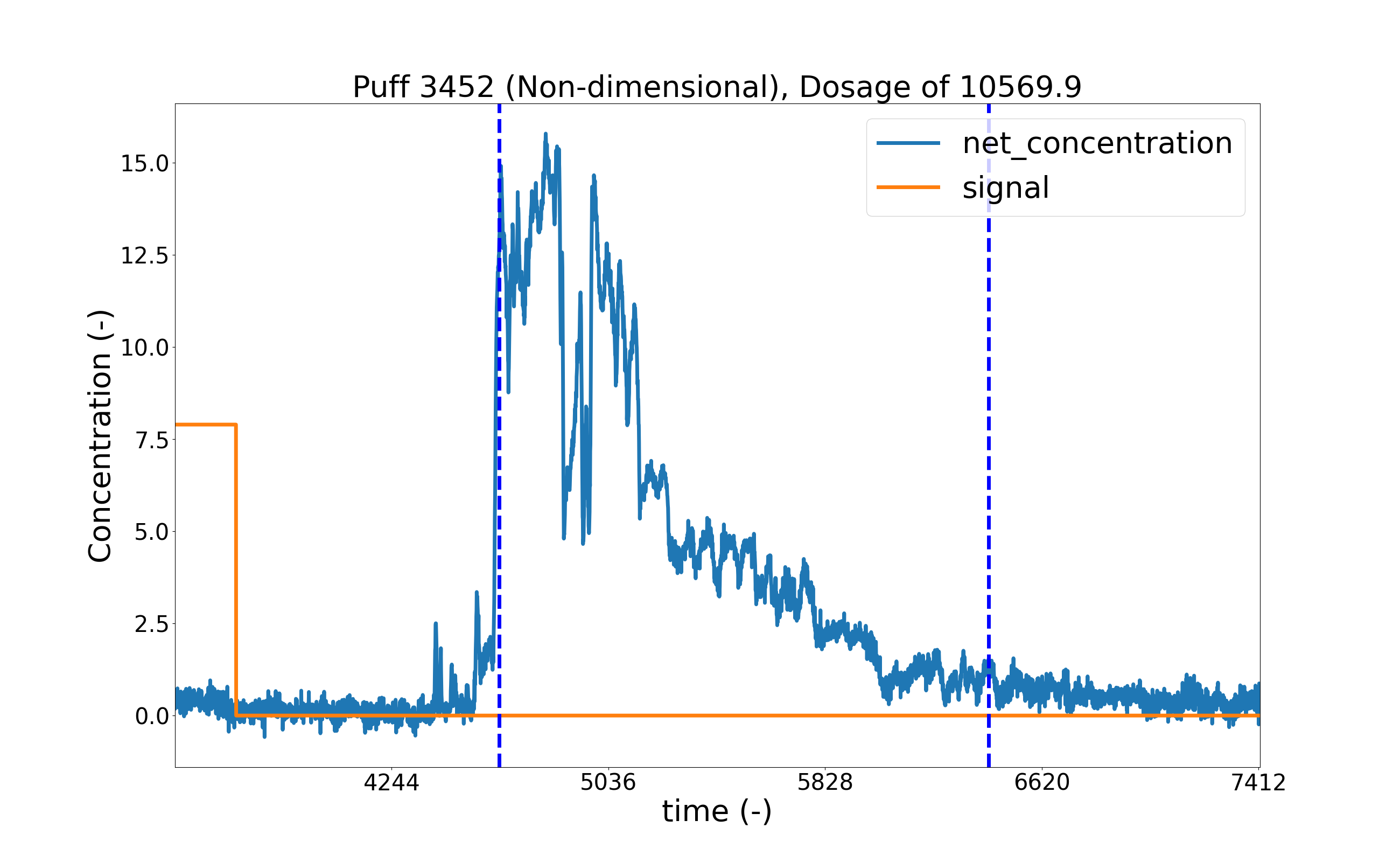


Figure 2: Puff plot of a single puff from sample time series, in full scale, model scale, and non-dimensional. Note that the puffs are labeled by the start time of their release.

signal (which is in binary form) is multiplied by the average peak concentration to make sure it is actually readable (see above). By default, the program plots only the first five puffs (in basic mode), or all puffs (in full mode). This can be also be manually changed (albeit only in CLI mode) by modifying the value for n\_puffs in line 289 (in basic mode) of 291 (full mode) of the python script, to either the desired number of puffs, or to ‘all’ to plot all puffs. Note that one of the main purposes of the existence of basic and full mode is to make the manual specification of the number of plots redundant.

Additionally, the program plots a plot similar to the puff plots (e.g. Figure 2), but for a characteristic ‘mean puff.’ This ‘mean puff’ is generated by linearly averaging the net concentration over all puffs in the dataset. In order to do this, the puff time series are truncated to the length of the shortest puff. This should not usually have a substantial impact on the data, except in rare cases where the tracer gas is released at irregular intervals (i.e. the puffs all have different lengths). In this case, the concentration data obtained after the end of the shortest puff will simply be left out of the analysis. Similarly to the regular puff plots, the mean puff plot also includes a characteristic arrival time and leaving time. However, the arrival time and the leaving time plotted on the mean puff plot are not the arrival and leaving time of the mean puff itself, but instead are obtained by calculating the arithmetic mean of all arrival and leaving times in the dataset. An average ‘peak time’, between the arrival and leaving time, is also included in the mean puff plot. By using the average of the characteristic arrival, leaving, and peak times, instead of the arrival, leaving, and peak times of the average puff, an insight of how close the puffs are to being normally distributed can be obtained (if the puffs were all perfectly normally distributed at every point in time, the average arrival, peak, and leaving time, would correspond exactly to the arrival, peak, and leaving time of the average concentration signal. Further, by setting the variable ‘dist’ to one (also in line 228) of the python script, the program will additionally plot the 10th and 90th percentile curve of the puffs. This gives a more quantitative insight into the distribution of the puffs at any given point in time.

Additionally, in the puff release mode, the program also outputs a set of convergence analysis plots. For this, the data is grouped into a set of ensembles, with varying ensemble size, and for each ensemble an ensemble mean is calculated. Note that for each ensemble size there exists as many ensembles as there are puffs. This analysis is performed separately for each variable (arrival time, ascent time, descent time, leaving time, peak concentration, and peak time). A more detailed description of the ensemble analysis and how it works can be found in Anne Philipp’s Bachelor Thesis (2010). The convergence analysis plots then effectively show the ensemble means plotted as a function of ensemble size. Ideally, the convergence analysis plots should have a funnel shape to them, with the top end of the funnel being on the left side of the plot, and the bottom of the funnel being at the right side of the plot.

Lastly, the program also outputs a set of frequency distribution plots. For this, the data are once again grouped into ensembles as for the convergence analysis plots, and the for each ensemble size the ensemble means are grouped into a set number of discrete classes, the frequencies (relative) of which are then output as a histogram. The number of classes varies as a function of ensemble size, and is calculated using a commonly used algorithm, described in more detail in Anne Philipp’s Bachelor Thesis (2010). Note that the analysis is once again performed separately for each of the variables listed above. Note that since the number of ensemble sizes equals the size of the data (in this case the number of puffs), the frequency analysis generates an extremely large number of plots (more precisely 6 times the number of puffs). For a reasonable number of puffs (easily 100 or more) this means that the frequency analysis can easily generate several thousand plots. For this reason, the frequency distribution plots are saved in a separate subfolder inside the data directory, titled ‘Frequency\_Distribution.’

1. **Remarks**

This program is still under development, but was initially developed by Johannes Fischer in the boundary layer wind tunnel working group under the supervision of Bernd Leitl at the Meteorological Institute at the University of Hamburg in 2019. The program is effectively an extension of the windtunnel package by Benyamin Schliffke (2018), some parts (where noted) are also based on an old analysis Program by Anne Philipp, developed for her Bachelor Thesis (2010). Many thanks also go to Frank Harms and Kerstin Surm for providing valuable feedback for improving the program. Some functions were also specifically developed by request from various members of the aforementioned working group (Helen Stricker, Rike Wachsmann, Frank Harms, and Kerstin Surm, among others). If you encounter any errors or have any other feedback please do not hesitate to contact the developers at [johannes.karl.fischer@gmail.com](mailto:johannes.karl.fischer@gmail.com).