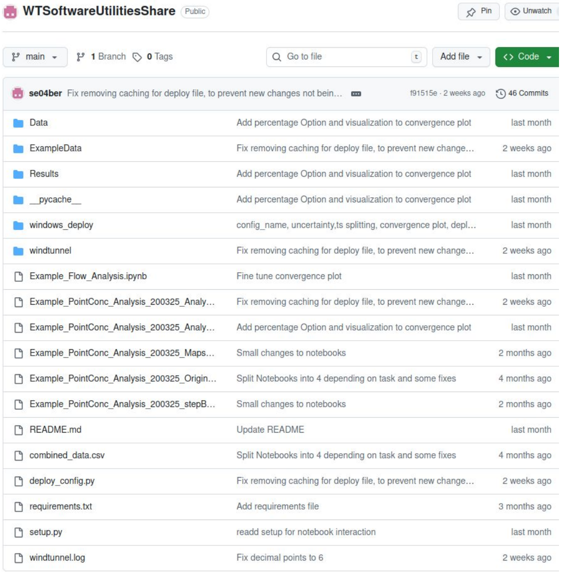
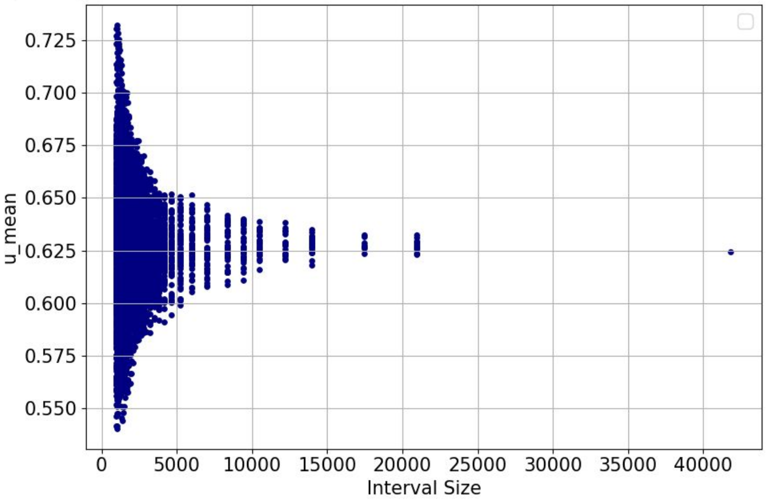
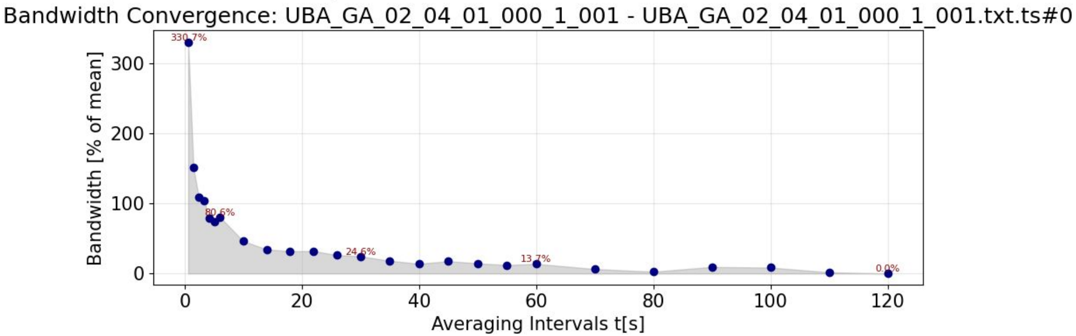
Windtunnel Software (+ UtilitiesShare-Repo)

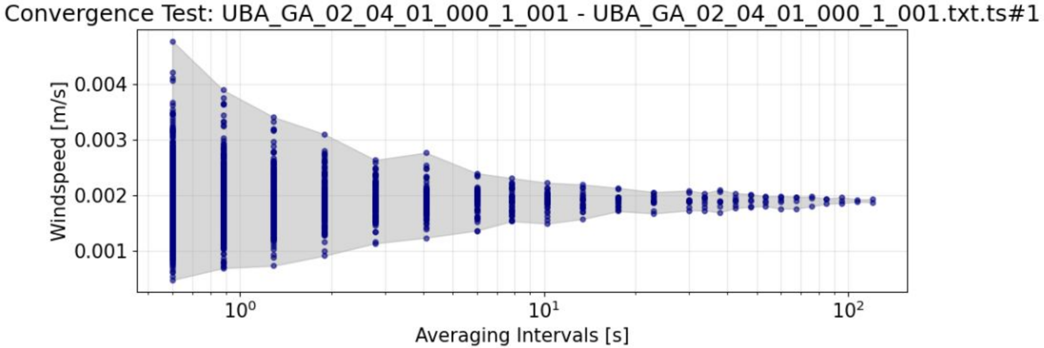
Short overview on
Features, Logic and Settings
and use of Repository



Convergence plots

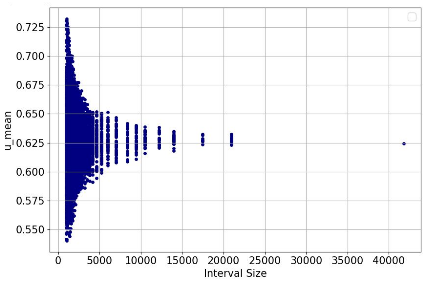






Flow_Analysis Notebook

- **Convergence Test:**
  Bandwidth of means for different averaging intervals



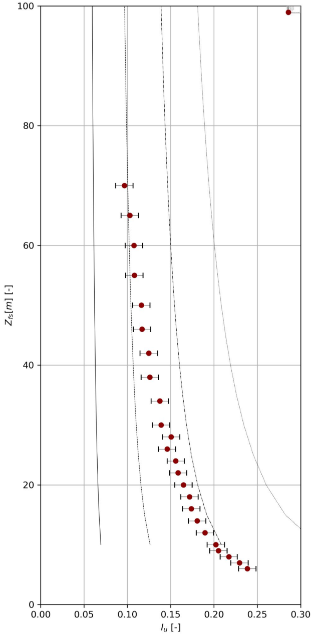
- **Vertical/Horizontal Wind Profiles:**
  Mean velocity(U) and turbulence intensity (Iu,Iv,Iw) vertically

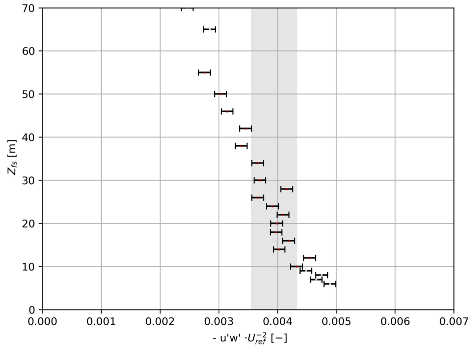
- **Boundary Layer Fitting:**
  Power law exponent α and roughness length z0

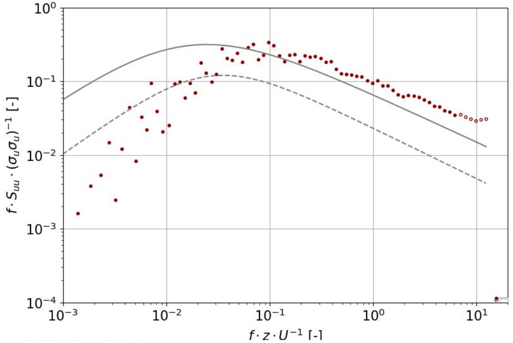
- **Turbulence Statistics:**
  Reynolds stresses vertically

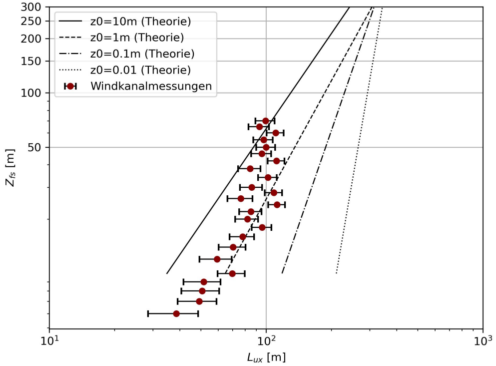
- **Integral Length Scales:**
  Lux from autocorrelation

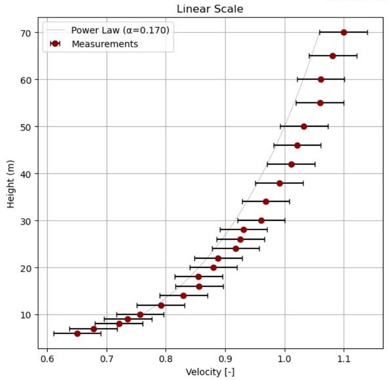
- **Spectral Analysis:**
  Power spectral density of velocity fluctuations

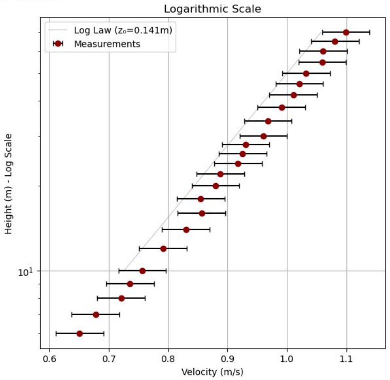








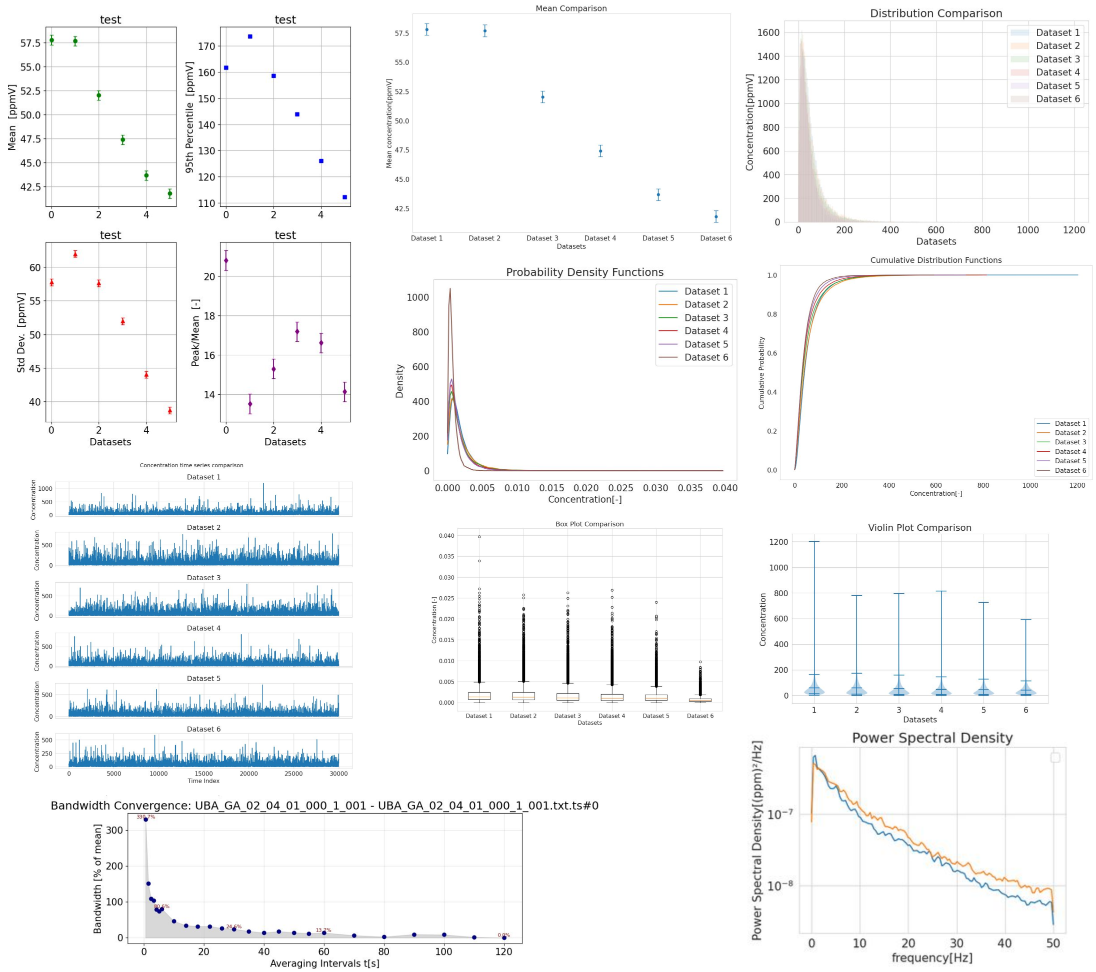




Concentration analysis Notebook

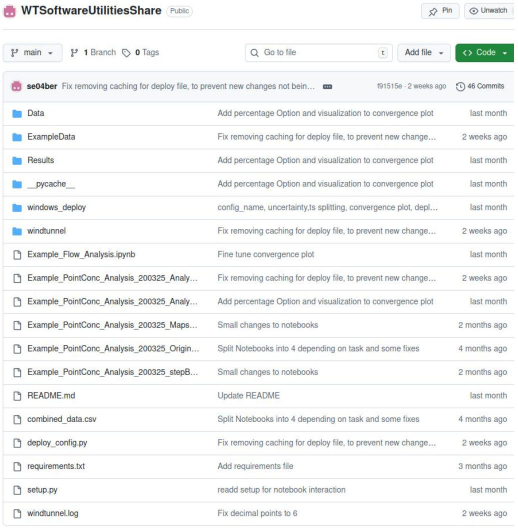
- All raw Ts files read-in into PointConc-object, postprocessing logic to (c_net[ppmV],c[-],c_fs[ppmV]) saved into array conc_ts[prefix][file]

- **Basic overview and distribution plots:** (Mean,Stdev, 95pecentile, Peak/Mean)-Table,Full ts plot, Means, Pdfs,Cdfs, Histograms, Boxplot, ViolinPlot



Next: Your Analysis (..) :)

Repository structure



- **Main Logic: package windtunnel/**
  concentration/ (PointConcentration.py, PuffConcentration.py)
  flow/
  plots/, grid/
  timeseries.py

- **Deployment files**
  (deploy_config.py - Jupyterhub server connection)
  windows_deploy/, requirements.txt)

- **Jupyter Notebooks**
  Flow_Analysis.ipynb - Interactive plotting of routine wind flow configuration check plots
  Example_PointConc_Analyis.ipynb - Interactive reading-in,postprocessing and plotting of concentration
  (Analysis, Analysis_deploy, stepByStep, Map) (Local machine version, JupyterHub version, Learning version, Map plotting version)

- **Data Folder**
  Data/InputData/ - Raw measurement time series
  Data/ParameterFiles/ - CSV files with ambient/(configuration) conditions/settings
  Results/ - Usually default results output folder
  ExampleData/ - Example time series files for testing

Deployment choices

1.Linux/Mac/Windows: **Quick-Start Option/Manual**	2.Windows: **Quick-Start Option/Manual**	3.Linux/Mac/Windows: **Quick-Start Option/Manual**
Use the notebook online in the JupterHub UUH server No python package local installations needed, but no change of windtunnel folder possible	Install package locally, but with use of helper scripts In two clicks, creates a clean environment "WTConc2", installs python dependencies, and runs the notebooks	Install package locally, manually (Clone repo manually and create a clean environment "WTConc2", install python dependencies, and run the notebooks manually) Git needed)

Quick Start Options

Option 1: Jupyter Server (Recommended for Students)

Best for: Using the notebook on a (f.e. the UHH) Jupyter Notebook online server without installing the full package.

Requirements: Jupyter Notebook server access

Steps:

1. Upload two files to your Jupyter server:
   - `Example_PointConc_Analysis_200325_Analysis_deploy.ipynb`
   - `deploy_config.py`
2. Open the notebook and run the first cell. The notebook will:
   - Automatically install the `windtunnel` package from GitHub
   - Create the necessary folder structure (`Data/InputData/` , `Data/ParameterFiles/` , `Results/`)
   - Download example data

Option 2: Full Installation: Windows Local using scripts

Best for: Installing, running and maybe editing or contributing to the full repository locally on Windows. When changes in the project folder should be made. Creates a clean environment "WTConc2" and runs notebooks in it with two clicks using helper scripts.

Requirements:

- Python >3.6 installed
- (Git installed (or download repository manually as ZIP from GitHub))

Steps:

1. **First-time setup** (run once):

   ```
   Run windows_deploy\setup_WTConc2.bat
   ```

   This creates a virtual environment `WTConc2` and installs all dependencies from requirements.txt.

2. **Start Jupyter Notebook** (every time you want to use it):

   ```
   Run windows_deploy\Start_WTConc2_Notebook.bat
   ```

3. Open any notebook f.e. `Example_PointConc_Analysis_200325_Analysis.ipynb` in the opened jupyter folder tree.

Option 3: Full Installation: Manually

Best for: Not Windows users who want do install, run and maybe edit the full project folder logic locally, contribute etc. Only the pip packages in requirements.txt need to be set up.

Requirements:

- Python >3.6 installed
- Git installed

Steps:

1. **Clone the repository:**

```
git clone https://github.com/se04ber/WTSoftwareUtilitiesShare.git
cd WTSoftwareUtilitiesShare
```

2. **Create a virtual environment:**

```
python -m venv WTConc2
```

3. **Activate the environment:**

   - **Windows:**

   ```
   WTConc2\Scripts\activate
   ```

   - **Linux/Mac:**

   ```
   source WTConc2/bin/activate
   ```

4. **Install dependencies:**

```
pip install -r requirements.txt
```

General structure of the notebooks

1. **Cell 0(Deploy): deploy setup and Data-path setup**
2. Cell 1: Paths and settings
3. Cell 2: Default csv parameter file values
4. Cell 3: Main Logic
5. Cell 4: Output testing
6. Cell 5: Bandwidth Convergence plot
7. Cell 6-: Basic Distribution analysis plots or flow plots
8. Your analysis and plots :)

JupyterHub version

deploy_config.py	16d ago
Example_PointC...	13d ago

```python
# Windtunnel Package Setup
from deploy_config import install_windtunnel, verify_installation

# Install and verify windtunnel package
if not install_windtunnel():
    print("❌ Installation failed. Please check your internet connection.")
else:
    verify_installation()
```

```
📦 Installing windtunnel package from GitHub...
Defaulting to user installation because normal site-packages is not writeable
Collecting git+https://github.com/se04ber/WTSoftwareUtilitiesShare.git
  Cloning https://github.com/se04ber/WTSoftwareUtilitiesShare.git to /tmp/pip-req-build-b3034m2_
  Running command git clone --filter=blob:none --quiet https://github.com/se04ber/WTSoftwareUtilitiesShare.git /tmp/pip-req-build-b3034m2_
  Resolved https://github.com/se04ber/WTSoftwareUtilitiesShare.git to commit f91515e393e77c883ba29b82ed21b763acc78349
  Preparing metadata (setup.py): started
```

```python
# Data source configuration
USE_GITHUB_EXAMPLE_DATA = True #False #True  # Set to False to use local data
#Else use your own Data in Folder Data/(your_data_folder)/measurement_prefix*, ...
DATA_FOLDER_NAME = "UBA Konvergenz"           #name of folder inside Data/InputData/ f.e. Umrechnung zur Kontrolle /
PARAMETER_FILE_NAME = "ambient_conditions.csv" #name of parameterFile inside ParameterFiles f.e. ambient_conditions_.UBA_GA.csv
MEASUREMENT_PREFIX = "your_measurement_prefix" #Prefix of all time series files inside DATA_FOLDER_NAME f.e. UBA_GA_02_04_01_000_1_00

# Data Setup and Configuration
from deploy_config import setup_folder_structure, setup_github_data, setup_local_data
# Setup folder structure and data
base_dir, data_dir, input_dir, param_dir, results_dir = setup_folder_structure()
if USE_GITHUB_EXAMPLE_DATA:
    path_dir, path, csv_file, output_path, namelist = setup_github_data(input_dir, param_dir, results_dir)
else:
    # Local data configuration
    DATA_FOLDER_NAME = DATA_FOLDER_NAME
    PARAMETER_FILE_NAME = PARAMETER_FILE_NAME
    MEASUREMENT_PREFIX = MEASUREMENT_PREFIX
    path_dir, path, csv_file, output_path, namelist = setup_local_data(
        input_dir, param_dir, results_dir, DATA_FOLDER_NAME, PARAMETER_FILE_NAME, MEASUREMENT_PREFIX
    )
print(f"✅ Setup complete! Data path: {path}, Output: {output_path}")
```

```
📁 Setting up folder structure...
✅ Created directory: /home/jovyan/Data
✅ Created directory: /home/jovyan/Data/InputData
✅ Created directory: /home/jovyan/Data/ParameterFiles
✅ Created directory: /home/jovyan/Results
🌐 Using GitHub example data
📥 Downloading example data from GitHub...
```

General structure of the notebooks

1. Cell 0(Deploy): deploy setup and Data-path setup
2. **Cell 1: Paths and settings**
3. Cell 2: Default csv parameter file values
4. Cell 3: Main Logic
5. Cell 4: Output testing
6. Cell 5: Bandwidth Convergence plot
7. Cell 6-: Basic Distribution analysis plots or flow plots
8. Your analysis and plots :)

Paths and settings

- path_dir = “(…)/WTSoftwareUtilitiesShare”
- path =“(…)/InputData/specificFolder/”
- output_path = “(..)/ExampleData/Results/”
- namelist = [“UBA_GA_001”]
- csv_file = “UBA_parameters.csv”

- parameters_PerFolder = False/True
- full_scale = “ms”/”fs”/”nd”

- osType = "Linux" / “Windows” #For Path
- outputName = None #Default: ts name
- saveTs=True,
- saveAvg=True,
- saveStats=True
- saveCombined=True
- combinedFileName="combined_file_nora.csv"
- base_path=Files/Point_Data_stats/UBA_thesis
- saveAll=True

```
#INPUT
path_dir = "/home/sabrina/Desktop/Schreibtisch/Arbeit_2025/WTSoftwareUtilitiesShare"
#Path to your input data
#path = f"{path_dir}/ExampleData/InputData/Concentration/"
path = f"{path_dir}/ExampleData/InputData/Beispiel Umrechnung zur Kontrolle/"
#path = f"/home/sabrina/Schreibtisch/Arbeit_2025/Nora_test/"
# Name of your measurement files prefix
#namelist = ['BFS_BD3_MP01_000']
namelist = ['UBA_GA_02_04_01_000_1_001'] #['UBA_thesis']
#Path to your output folder for average files and plots
output_path = f"{path_dir}/ExampleData/Results/"
#For PointData for the functions to work the columns of the file should be: time, wtref, slow_FID, fast_FID, open_rate
#(See manual for the description of the variables)
#Name of csv file which contains ambient conditions data. Multiple diff. ambient conditions for diff datasets can be read-in at ones
#If no file given or configuration wrong, the program ressorts to try reading-in given values manually below.
#csv_file='ambient_conditions.csv'
#csv_file= f"{path_dir}/ExampleData/ParameterFiles/ambient_conditions.csv"
csv_file= f"{path_dir}/ExampleData/ParameterFiles/ambient_conditions_.UBA_GA.csv"
#csv_file= f"{path}ambient_conditions_point1.csv"
parameters_PerFolder = False

#Variables and Parameters set for all ts, if no ambient_conditions.csv file overgiven
#If at the end calculate entdimensionalised or full scale transform quantities
#Default: nd:entdimensionalise, ms:model scale, fs:full scale.
full_scale='ms'
#Postprocessing before analysis
applyPostprocessing=True
averageInterval=60 #s  #Interval to downaverage raw time series to before analysis
measurementFreq=0.005 #Time series frequency #For now only for static case implemented
averagingColumns=["net_concentration"] #Columns to average dow
#Saving settings: (output_path for path)
osType = "Linux" #Windows  #For Path
outputName = None #Default: ts name

saveTs=True    #Only save time series of concentration quantities to separat files
saveAvg=True #Save average of ts of concentration quantities to separat files
saveStats=True #Save averages, percentile95/5, peak2mean of ts of concentration quantities to separat files
saveCombined=True #Save all averages and statistics for all files to one combined file
combinedFileName="combined_file_nora.csv"
base_path=None #base_path = output_path + "Files/Point_Data_stats/UBA_thesi/" # Base path for getting files for combined files, if None

saveAll=True #Sets saveTs, saveAvg and saveStats, saveCombined to True, saving ts, averages, statistics and combined file
```

Data/InputData/UBA/
UBA_GA_001_ts#1
UBA_GA_001_ts#2
UBA_GA_001_ts#3
….
Data/ParameterFiles/
UBA_parameters.csv

General structure of the notebooks

1. Cell 0(Deploy): deploy setup and Data-path setup
2. Cell 1: Paths and settings
3. **Cell 2: Default csv parameter file values**
4. Cell 3: Main Logic
5. Cell 4: Output testing
6. Cell 5: Bandwidth Convergence plot
7. Cell 6-: Basic Distribution analysis plots or flow plots
8. Your analysis and plots :)

Parameterfile

- x/y/z_source =(?,?,?) [mm]
- x/y/z_measure=(?,?,?) [mm]
- mass_flow_controller = 0.300 [l/h]
- calibration_curve=1.0 [-]
  calibration_factor=0.0 [-]
- gas_name=”C12” [-]
- gas_factor=0.5 [-]
- mol_weight=29.0 [g/mol]
- pressure=101426.04472 [Pa]
- temperature=23 [°C]
- scale=400 [Model/Reality]
- scaling_factor=0.5614882 [Model/Reality]
- ref_length= 1/400
- ref_height= 100/400
- full_scale_wtref= 10 [m/s]
- full_scale_flow_rate=0.002 [kg/s]
- full_scale_temp=20 [°C]
- full_scale_pressure=101325 [Pa]
- config_name=”test” [-]

```
#Example file/Default environment values if no csv_file found:

#Source location  [mm]
x_source=0
y_source=0
z_source=0
#Source mass flow controller, calibration settings
mass_flow_controller=0.300 #0.600#Stickstoffdurchflussregler #[l/h]*1/100 #'X'  #Controller(settings) used, just a name placeholder for orient
#If calibration performed on a controller, corrects actual max. flow capacity of controller
calibration_curve=1.0     #0.3     #0.3 oder 3
calibration_factor=0 #1      #
#Gas characteristics
gas_name='C12'           #Just placeholder name variable for orientation, not used for anything
gas_factor=0.5   #[-]    #!!! Needs to be calculate/specificate f.e. if gas changes
mol_weight=29.0 #28.97 #Air [g/mol]
#Measurement location [mm]
x_measure=1020 #855.16
y_measure= 0    #176.29
z_measure= 5     #162
#Surrounding conditions
pressure=101426.04472        #1009.38  #[hPa] ->Pa
temperature=23             #23.5  #[°C]
#Model to Reality scaling
scale=400                    #250      #Model/Reality
scaling_factor=0.5614882             #0.637       #USA1 to selected ref pos.?
ref_length=1/400             #1/250           #Lref
ref_height=100/400           #None            #Href
#Full Scale Parameters
full_scale_wtref=10            #6         #Uref_fullscale
full_scale_flow_rate=0.002     #Q_amb[kg/s]?   #0.5   #Qv_fullscale
full_scale_temp=20             #[°C]
full_scale_pressure=101325     #[Pa]
#Q_ambient[kg/s] ->  Q[m³/s]=Q[kg/s]*R*T/(M*p)
#Variable wdir for wind direction. To be implemented in future. ##
#wdir=0
#Variable axis_range. Reserved for future implementation of axis range specification,
#analogously to puff mode
#axis_range='auto'
```

```
,UBA_GA_02_04_01_000_1_001.txt.ts#0,UBA_GA_02_04_01_000_1_001.txt.ts#1,UBA_GA_02_04_01_000_1_001.txt.ts#2,UBA_GA_02_04_01_000
x_source,0.0,0.0,0.0,0.0,0.0,0.0
y_source,0.0,0.0,0.0,0.0,0.0,0.0
z_source,0.0,0.0,0.0,0.0,0.0,0.0
x_measure,37.5,50.0,62.5,75.0,87.5,100.0
y_measure,0.0,0.0,0.0,0.0,0.0,0.0
z_measure,10.0,10.0,10.0,10.0,10.0,10.0
pressure,101768,101768,101768,101768,101768,101768
temperature,23.0,23.0,23.0,23.0,23.0,23.0
calibration_curve,None,None,None,None,None,None
mass_flow_controller,0.300,0.300,0.300,0.300,0.300,0.300
calibration_factor,1,1,1,1,1,1
scaling_factor,0.656868699877201,0.656868699877201,0.656868699877201,0.656868699877201,0.656868699877201,0.656868699877201
scale,200,200,200,200,200,200
ref_length,0.005,0.005,0.005,0.005,0.005,0.005
ref_height,None,None,None,None,None
gas_name,C12,C12,C12,C12,C12,C12
mol_weight,28.97,28.97,28.97,28.97,28.97,28.97
gas_factor,1.0,1.0,1.0,1.0,1.0,1.0
full_scale_wtref,5.0,5.0,5.0,5.0,5.0,5.0
full_scale_flow_rate,0.5,0.5,0.5,0.5,0.5,0.5
full_scale_temp,20,20,20,20,20,20
full_scale_pressure,101325,101325,101325,101325,101325,101325
config_name,"test","test","test","test","test","test"
```

General structure of the notebooks

1. Cell 0(Deploy): deploy setup and Data-path setup
2. Cell 1: Paths and settings
3. Cell 2: Default csv parameter file values
4. **Cell 3: Main Logic**
5. Cell 4: Output testing
6. Cell 5: Bandwidth Convergence plot
7. Cell 6-: Basic Distribution analysis plots or flow plots
8. Your analysis and plots :)

Main Logic - mass flow rate

- model scale
  ambient mass flow rate Q

- Qcontroller = Maximum flow rate capacity of gate (150l/h)

- Multiplied with open_rate[1-10] of gate (ts file) and gas factor and ambient conditions(T,p)

$$\dot{Q}_{\text{amb}} = \frac{\text{open\_rate} \cdot 10 \cdot f_{\text{gas}} \cdot \dot{Q}_{\text{controller}}}{100} \cdot \frac{T_K \cdot p_{\text{std}}}{p \cdot T_{\text{std,K}}}$$

- $\dot{Q}_{\text{amb}}$ – ambient mass flow rate [l/h]
- open_rate – valve opening [0-10], multiplied by 10 to get percentage
- f_{gas} – gas correction factor [-] (e.g., for different tracer gases)
- $\dot{Q}_{\text{controller}}$ – maximum flow rate of mass flow controller [l/h]
- T_K – ambient temperature [K]
- p – ambient pressure [Pa]
- $T_{\text{std,K}}$ – standard temperature = 273.15 K
- p_{std} – standard pressure = 101325 Pa

Main Logic - Net Concentration

- net_concentration C [ppmV]

$$\boxed{C_{\text{net}} = \text{fast\_FID} - \text{slow\_FID}}$$

- slow_FID - total concentration measured by fast flame ionization detector
- fast_FID - background concentration [ppmV]

Main Logic - Entdimensionalised Concentration

- ❑ Entdim. concentration C [-]
- ❑ Comparing measurements at different scales and conditions

$$\boxed{C^* = \frac{C_{\text{net}} \cdot \overline{U}_{\text{ref}} \cdot L_{\text{ref}}^2}{\dot{Q}_{\text{amb}}}}$$

- C^* – dimensionless concentration [-]
- C_{net} – net concentration [ppmV], converted to volume fraction by 10^{-6}
- $\overline{U}_{\text{ref}}$ – mean reference wind speed [m/s]
- L_{ref} – reference length (model scale) [m]
- $\dot{Q}_{\text{amb}}$ – ambient flow rate [l/h], converted to [m³/s] by $(1000 \cdot 3600)^{-1}$

$$C^* = \frac{\left(\frac{C_{\text{net}}}{10^6}\right) \cdot \overline{U}_{\text{ref}} \cdot L_{\text{ref}}^2}{\left(\frac{\dot{Q}_{\text{amb}}}{1000 \cdot 3600}\right)}$$

Main Logic

```
conc_ts[name][file] = wt.PointConcentration.from_file(path + file)

conc_ts[name][file].ambient_conditions(x_source=x_source, y_source=y_source, z_source=z_source,
                                       x_measure=x_measure, y_measure=y_measure, z_measure=z_measure,
                                       pressure=pressure,
                                       temperature=temperature,
                                       calibration_curve=calibration_curve,
                                       mass_flow_controller=mass_flow_controller,
                                       calibration_factor=calibration_factor,
                                       config_name=config_name)

#Set read-in scaling, gas and full scale information to internal class variables
print("Store information into PointConcentration class objects array")
conc_ts[name][file].scaling_information(scaling_factor=scaling_factor,
                                        scale=scale,
                                        ref_length=ref_length,
                                        ref_height=ref_height)
conc_ts[name][file].tracer_information(gas_name=gas_name,
                                       mol_weight=mol_weight,
                                       gas_factor=gas_factor)
conc_ts[name][file].full_scale_information(full_scale_wtref=full_scale_wtref,
                                           full_scale_flow_rate=full_scale_flow_rate,
                                           full_scale_temp=full_scale_temp,full_scale_pressure=full_scale_pressure)

#Calculate mass flow rate, net concentration and dimensionalise concentration
print("Do main calculations")
conc_ts[name][file].convert_temperature()
conc_ts[name][file].calc_wtref_mean()

conc_ts[name][file].calc_model_mass_flow_rate(usingMaxFlowRate="True",applyCalibration="False")
conc_ts[name][file].calc_net_concentration()

#conc_ts[name][file].clear_zeros()  #Remove values net_concentration =< 0 from dataset !noise
conc_ts[name][file].calc_c_star()

conc_ts[name][file].calc_full_scale_concentration() #Try
```

Main Logic

```
conc_ts[name][file] = wt.PointConcentration.from_file(path + file)

conc_ts[name][file].ambient_conditions(x_source=x_source, y_source=y_source, z_source=z_source,
                                       x_measure=x_measure, y_measure=y_measure, z_measure=z_measure,
                                       pressure=pressure,
                                       temperature=temperature,
                                       calibration_curve=calibration_curve,
                                       mass_flow_controller=mass_flow_controller,
                                       calibration_factor=calibration_factor,
                                       config_name=config_name)

#Set read-in scaling, gas and full scale information to internal class variables
print("Store information into PointConcentration class objects array")
conc_ts[name][file].scaling_information(scaling_factor=scaling_factor,
                                        scale=scale,
                                        ref_length=ref_length,
                                        ref_height=ref_height)
conc_ts[name][file].tracer_information(gas_name=gas_name,
                                       mol_weight=mol_weight,
                                       gas_factor=gas_factor)
conc_ts[name][file].full_scale_information(full_scale_wtref=full_scale_wtref,
                                           full_scale_flow_rate=full_scale_flow_rate,
                                           full_scale_temp=full_scale_temp,full_scale_pressure=full_scale_pressure)

#Calculate mass flow rate, net concentration and dimensionalise concentration
print("Do main calculations")
conc_ts[name][file].convert_temperature()
conc_ts[name][file].calc_wtref_mean()

conc_ts[name][file].calc_model_mass_flow_rate(usingMaxFlowRate="True",applyCalibration="False")
conc_ts[name][file].calc_net_concentration()

#conc_ts[name][file].clear_zeros()  #Remove values net_concentration =< 0 from dataset !noise
conc_ts[name][file].calc_c_star()

conc_ts[name][file].calc_full_scale_concentration() #Try
```

Main Logic

```python
conc_ts[name][file] = wt.PointConcentration.from_file(path + file)

conc_ts[name][file].ambient_conditions(x_source=x_source, y_source=y_source, z_source=z_source,
                                       x_measure=x_measure, y_measure=y_measure, z_measure=z_measure,
                                       pressure=pressure,
                                       temperature=temperature,
                                       calibration_curve=calibration_curve,
                                       mass_flow_controller=mass_flow_controller,
                                       calibration_factor=calibration_factor,
                                       config_name=config_name)

#Set read-in scaling, gas and full scale information to internal class variables
print("Store information into PointConcentration class objects array")
conc_ts[name][file].scaling_information(scaling_factor=scaling_factor,
                                        scale=scale,
                                        ref_length=ref_length,
                                        ref_height=ref_height)
conc_ts[name][file].tracer_information(gas_name=gas_name,
                                       mol_weight=mol_weight,
                                       gas_factor=gas_factor)
conc_ts[name][file].full_scale_information(full_scale_wtref=full_scale_wtref,
                                           full_scale_flow_rate=full_scale_flow_rate,
                                           full_scale_temp=full_scale_temp,full_scale_pressure=full_scale_pressure)

#Calculate mass flow rate, net concentration and dimensionalise concentration
print("Do main calculations")
conc_ts[name][file].convert_temperature()
conc_ts[name][file].calc_wtref_mean()

conc_ts[name][file].calc_model_mass_flow_rate(usingMaxFlowRate="True",applyCalibration="False")
conc_ts[name][file].calc_net_concentration()

#conc_ts[name][file].clear_zeros()  #Remove values net_concentration =< 0 from dataset !noise
conc_ts[name][file].calc_c_star()

conc_ts[name][file].calc_full_scale_concentration() #Try
```

Main Logic

- ❑ Transform to scale
- ❑ Save files
(ts: c_net,c*,c_fs)
(avg: ..)
(stats: peak2Mean,
percentiles)

```python
#Transforming/Outputting data in full-scale, model scale, and non-dimensional
print("Transform scale")
if full_scale == 'ms':
    dict_conc_ts = conc_ts

elif full_scale == 'fs':
    dict_conc_ts = conc_ts_fs
    dict_conc_ts[name][file].to_full_scale()

elif full_scale == 'nd':
    dict_conc_ts = conc_ts_nd
    dict_conc_ts[name][file].to_non_dimensional()
else:
    print(
        "Error: invalid input for full_scale. Data can only be computed in model scale (full_scale='ms'), full scale (full_scale='f
#Apply Postprocessing if overgiven
```

```python
for file in files:
    if(saveTs):
        if full_scale == 'ms':
            dict_conc_ts[name][file].save2file_ms(file, out_dir=output_path + folder)
        elif full_scale == 'fs':
            dict_conc_ts[name][file].save2file_fs(file, out_dir=output_path + folder)
        elif full_scale == 'nd':
            dict_conc_ts[name][file].save2file_nd(file, out_dir=output_path + folder)
        else:
            print(
                "Error: invalid input for full_scale. Data can only be computed in model scale (full_scale='ms'), full scale (full_s
            exit
        print("Created ts files including (net_concentration, entimendionsliased and full scale concentration)")
    if(saveAvg):
         #Saving averages to files under folder Point_Data_stats/
        #Averages of net_concentration,c_star and full_scale_concentration
         wt.check_directory(output_path + folder_avg)
         dict_conc_ts[name][file].save2file_avg(file, out_dir=output_path + folder_avg)
         print(f"Created avg files under {output_path + folder_avg}")
    if(saveStats):
        #Saving stats to files under folder Point_Data_avg/
        #Stats Full ausgabe: saveAvg Quantities + Percentile 95, percentile 5, peak2Mean of net_concentration_c_star and full_Scale_
         wt.check_directory(output_path + folder_stats)
         dict_conc_ts[name][file].save2file_fullStats(file, out_dir=output_path + folder_stats)
         print(f"Created stats files under {output_path + folder_stats}")

if(saveCombined):
    from windtunnel.concentration.utils import combine_to_csv2
    stats=True
    if(stats):
        file_type="stats"

    file_names = ["_stats_" + file for file in files]
    if(base_path==None):
        base_path = output_path + f"Files/Point_Data_{file_type}/{name[0:-1]}/"
```

General structure of the notebooks

1. Cell 0(Deploy): deploy setup and Data-path setup
2. Cell 1: Paths and settings
3. Cell 2: Default csv parameter file values
4. Cell 3: Main Logic
5. **Cell 4: Output testing**
6. **Cell 5: Bandwidth Convergence plot**
7. **Cell 6-: Basic Distribution analysis plots or flow plots**
8. **Cell (-,-): Your analysis and plots :)**

Next Cells

- Read into Help-Array for plotting, Overview on stats quantitatively
- Then: Called plotting functions with defined settings for each cell (basic distribution plots)
- Next Cells: your own project specific analysis and plots :)

```
DataPointsConc = []
#DataPointsConc = [ conc_ts[namelist[0]]["MyFileofInterest1.txt"], conc_ts[namelist[0]]["MyFileofInterest2.txt"]]
for i in range(len(files)): #Just visualising all
    data = conc_ts[namelist[0]][files[i]]
    DataPointsConc.append(data)
# Richtige Zeitserien laden
labels=[f"Dataset {i}" for i in range(0,len(DataPointsConc))]
plot_timeseries_with_stats(DataPointsConc,dimensionless=dimensionless,labels=labels,color=color)
```

```
 File: UBA_GA_02_04_01_000_1_001.txt.ts#0
c_star vorhanden
C_star shape: (30000,)
NaNs vorhanden: False
Min/Max: -0.30316299999999985 / 1202.017763
Mean: 57.80034634370001
Std: 57.766640161132536
Percentiles: {10: 12.215996500000001, 90: 120.00919130000001, 95: 161.68999759999997}
```

```
[15]: xLabel="Datasets"
yLabel="Concentration[ppmV]"
xLabel="Datasets"
yLabel="Concentration[ppmV]"
create_histogram(DataPointsConc,dimensionless="False",labels=None,xLabel=xLabel,yLabel=yLabel,xAchse=None,yAchse=None)
```