Open Access Educational Resources and Data Analysis Tools for Fire Simulations

Lukas Arnold [a,b]

[a] Institue for Advanced Simulation, Forschungszentrum Jülich, Jülich, Germany
[b] Chair of Computational Civil Engineering, University of Wuppertal, Wuppertal, Germany
[*] Corresponding author: `l.arnold@fz-juelich.de`, `arnold@uni-wuppertal.de`

BERGISCHE UNIVERSITÄT WUPPERTAL

JÜLICH Forschungszentrum

Lecture Notes on Fire Simulations

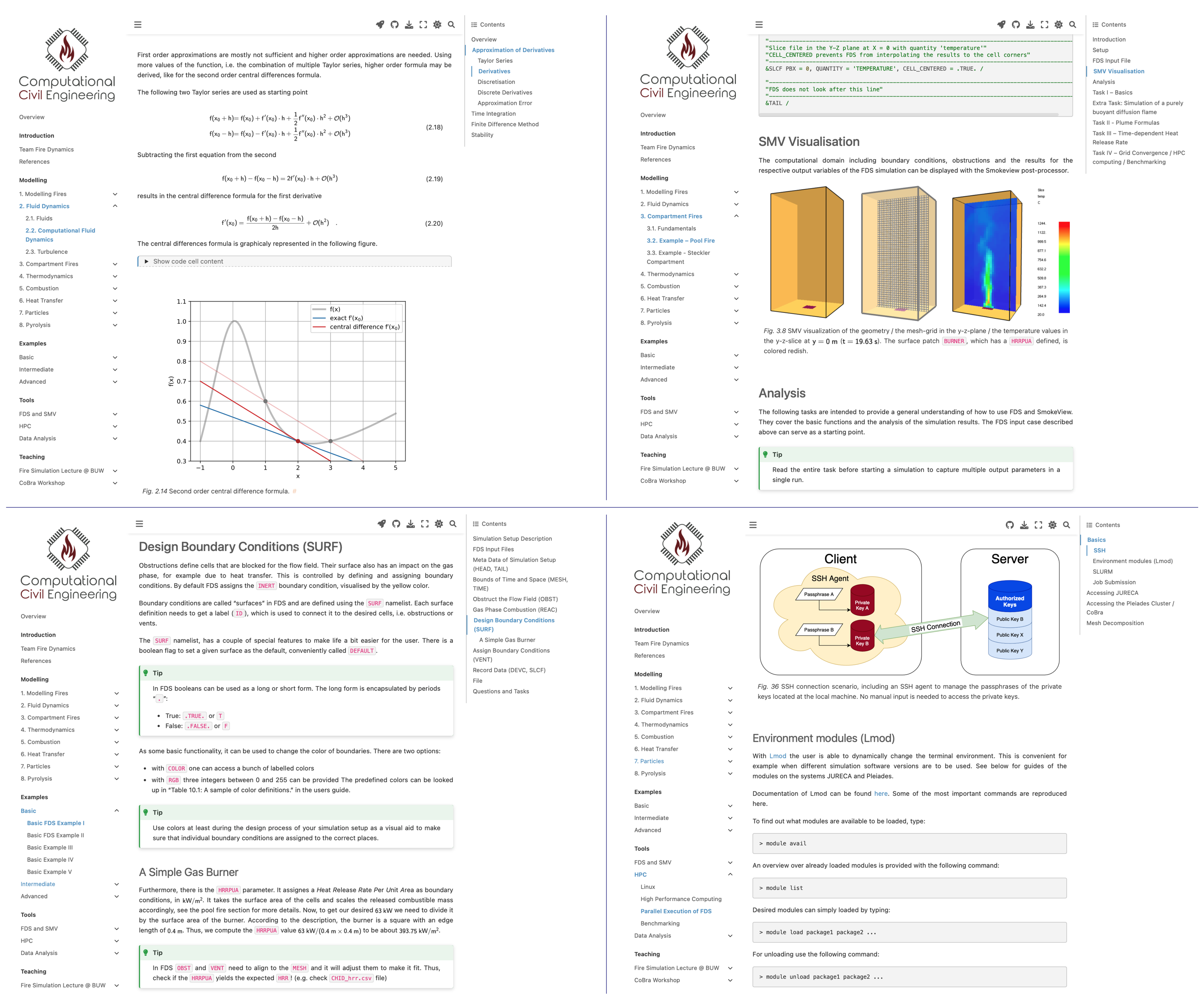
The lecture notes cover teaching material for (at least) 14 weeks, with 4 hours per week, as well as additional content and examples. We address the major phenomena (fluid dynamics, combustion, heat transfer, etc.), by first providing a fundamental introduction and then introducing the underlying modelling approaches. The models are demonstrated by simple examples as well as validation cases. Besides the introduction to the usage of FDS (Fire Dynamics Simulator), we cover topics like: Linux for beginners, running FDS on an HPC (high performance computing) system, and data analysis with Python.

Link Repo: `https://github.com/FireDynamics/LectureFireSimulation`
Link Build: `https://firedynamics.github.io/LectureFireSimulation`
Authors: Kristian Börger, Tristan Hehnen, Lilli Klein, Karen De Lannoye, Keyvan Najarian, Tássia Quaresma, Jan Vogelsang, My Linh Würzburger, Lukas Arnold

Content and Time Schedule

Week	Part I	Part II
1	Lecture Overview and Organisation	Fire Safety Science and Engineering
2	Fire Modelling Approaches	Linux and High Performance Computing
3	Fluid Dynamics	Data Analysis with Python
4	Computational Fluid Dynamics	Turbulence modelling
5	Fire Dynamics Simulator (FDS)	Compartment Fires
6	Pool Fire Example	Fundaments of Thermodynamics
7	Design Fire Example	Parallel Execution of FDS
8	Thermodynamics Examples	Analysis of FDS Data with Python
9	Combustion Modelling	Basics of Combustion
10	Combustion Examples	Advanced Analysis Examples
11	Heat Transfer Modelling	Fundamentals of Heat Transfer
12	Heat Transfer Examples	Pyrolysis Modelling
13	Verification and Validation	Simulation Project (Intro + Overview)
14	Recap, Thesis and Projects	Simulation Project (First Steps)

Impressions



Implementation

jupyter + JB + GitHub =

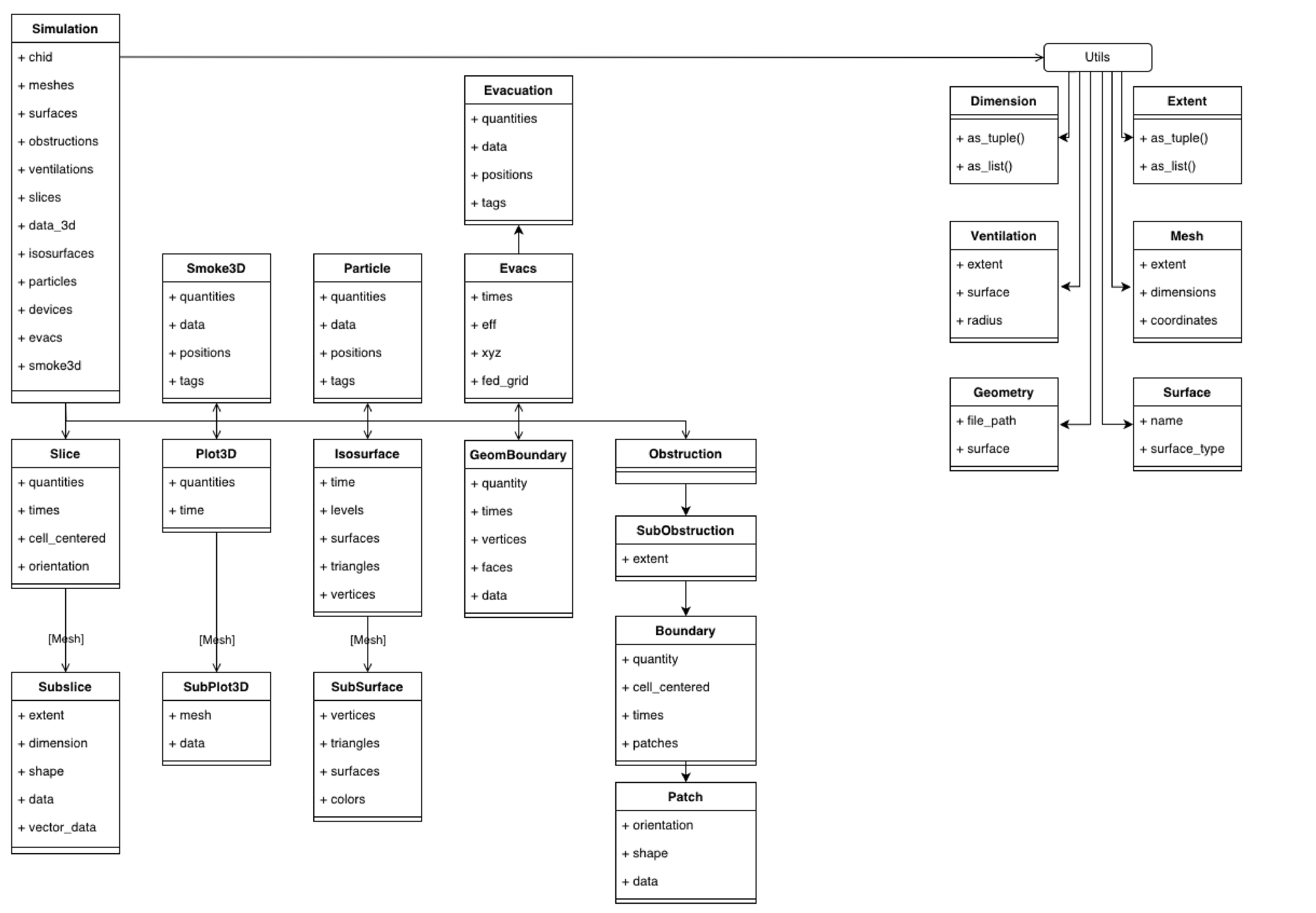
Tools used: Python and Markdown in Jupyter notebooks, HTML build used JupyterBook and the source and webpage are hosted on GitHub. The build environment and GitHub actions for automated deployment are included in the repository.

`fdsreader` Python module

To ease the analysis of simulation data computed with FDS, we have developed the tool `fdsreader`, a Python module available on GitHub and PyPI. The module assists users in efficiently loading and analysing FDS simulation data, thereby improving the accessibility and usability of fire simulation results. It exemplifies how fire simulations, underpinned by computer science principles, can be effectively interpreted and utilised. With growing interest in this tool, we have already conducted training sessions for practitioners to familiarise them with its use.

Link Repo: `https://github.com/FireDynamics/fdsreader`
Authors: Jan Vogelsang, Lukas Arnold

Data Structures

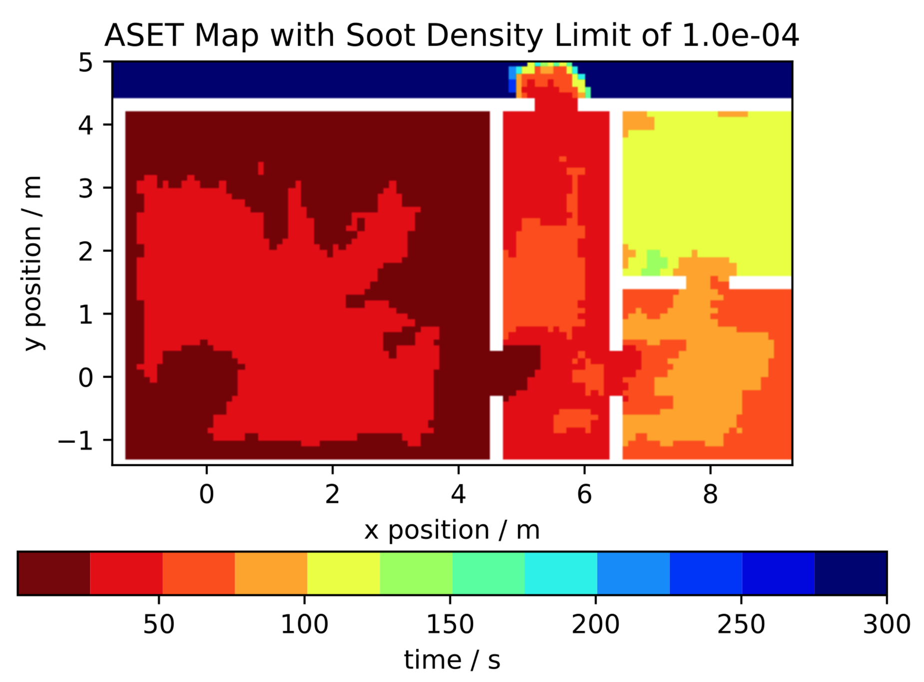


Application Example – ASET Map

The following example illustrates the approach to read in SLCF data and post process it. The resulting structure is a three dimensional array (two spatial and one temporal dimension). Here, we create an ASET map following the procedure based on Schröder et al. ("A map representation of the ASET-RSET concept", Fire Safety Journal 2020). To compute the map, the soot density is evaluated at a given height for each floor point in the compartment. The first violation of the tenability criterion fixes the time value of the ASET map.

```python
import fdsreader
import numpy as np

path_to_data = 'dir/containing/smv_file'
sim = fdsreader.Simulation(path_to_data)

# get the soot density slice, normal to z at 1.5m height
slc = sim.slices.get_by_id('SootDensityZ_1.5m')

# as the simulation is based on multiple meshes, a global
# data structure is created, walls are represented as
# non-valid data points, i.e. nan
slc_data = slc.to_global(masked=True, fill=np.nan)
```

```python
# set arbitrary values as tenability threshold
soot_density_limit = 1e-4

# create a map with max ASET as default value
aset_map = np.full_like(slc_data[0], slc.times[-1])

# set walls to nan
aset_map[np.isnan(slc_data[0,:,:])] = np.nan

# 1D loop over all array indices, ix is a two dimensional index
for ix in np.ndindex(aset_map.shape):

    # find spatialy local values which exceed the given limit
    local_aset = np.where(slc_data[:, ix[0], ix[1]] > soot_density_limit)[0]

    # if any value exists
    if len(local_aset) > 0:
        # use the first, i.e. first in time, as the local ASET value
        aset_map[ix] = slc.times[local_aset[0]]
```



Getting Started

1. **Installation:** `pip install fdsreader`
2. **Tutorials:** See lecture notes, section Tools/Data Analysis/fdsreader Module

Link to lecture notes

Acknowledgements

This work is partially funded within the CoBra project, which is funded by the German Federal Ministry of Education and Research under the project number 13N15497.

SPONSORED BY THE Federal Ministry of Education and Research

Link to `fdsreader`