

## Pre- / Postprocessing Tools

Summer School on Fire Dynamics Modeling 2017

Lukas Arnold

## Contents:

1. `fdsgeogen`
2. Python FDS data reader
3. Multivariate Analysis

## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions

## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions

# Motivation

## sensitivity analysis

- ▶ investigate impact of parameter multitude
- ▶ gain individual scenario independent system description
- ▶ input data for further probabilistic analysis

## automatisation

- ▶ generation of FDS input files with relative declaration
- ▶ work pipeline: define parameter - prepare input - run - analyse
- ▶ open source / easy to adopt / run on parallel computer

## Overview

`fdsgeogen` is a small collection of tools to ease automatisisation with FDS

- ▶ it is implemented in Python
- ▶ so far, works best on Linux/OSX systems

Main idea:

1. formulate parameter space
2. create FDS input files based on chosen parameter space
3. keep track of simulations to run and data to be analysed
4. run simulation in serial or parallel
5. analyse data

## Tools

`fdsgeogen` , or short `fgg`, comes with a small compilation of tools:

- ▶ `fgg_create`, parse XML file to generate a collection of FDS input files
- ▶ `fgg_run_serial`, runs all created FDS input files in serial
- ▶ `fgg_analyse`, analyses the computed data

`fdsgeogen` creates a few helper files to manage the pipeline, e.g.

- ▶ `fgg.subdirlist`, list of all created subdirectories
- ▶ `fgg.plot`, device data plotting instruction

## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions



## Pythonic XML syntax

- ▶ fdsgeogen uses XML as input syntax

---

```
<tag  at1="3.6" />
```

---

- ▶ all attributes are evaluated by Python and must be therefore valid Python types

---

```
<info  intarg="4"  floatarg="3.421"  strarg="'exercise'" />
```

---

- ▶ enclosing tags

---

```
<loop  var="i"  stop="10" >  
    ... loop body ...  
</loop>
```

---

## Expression evaluation

- ▶ all attributes are evaluated by the Python interpreter and therefor allow for all possible Python expressions

---

```
<info addition="4+6.5" />
```

---

- ▶ variables may be defined and used for later evaluation

---

```
<var a="4.6" />  
<info addition="4 + a" />
```

---

## How to run fdsgeogen on VM

1. fdsgeogen uses Python2 and the numpy module
2. you need to install those packages on the VM, go to the software manager and select python2-numpy to be installed
3. use

---

```
python2 <fgg_create_script> <xml_input_file>
```

---

---

```
python2 ../fgg/scripts/fgg_create.py e01.xml
```

---

## XML tags (I)

There exist a couple of fundamental tags to setup the fdsgeogen framework

- ▶ `<var>`, defines variables
- ▶ `<dbg>`, prints output to stdout, not to the FDS input file
- ▶ `<loop>`, allows loop definition

→ hands-on example 1

The handling of FDS input files is managed, by e.g.

- ▶ `<info>`, provides info about chid, subdirectories and file names
- ▶ `<dump>`, writes directly to the FDS input file
- ▶ `<input>`, writes FDS statements or imports selected data from existing FDS file
- ▶ `<para>`, parameter space definition

→ hands-on example 2

## XML tags (II)

FDS specific input can be handled by tags starting with `fds_*`, e.g.

- ▶ `<fds_reac>`, defines the REAC line
- ▶ `<fds_mesh>`, defines the MESH line
- ▶ `<fds_surf>`, defines the SURF line

→ hands-on example 3

Additionally there are a couple of combined commands, e.g.

- ▶ `<bounded_room>`, defines a room, including a (split up) mesh with walls
- ▶ `<fire>`, defines simple types of fires
- ▶ `<devc>`, defines (multiple) devices together with plotting options

→ hands-on example 4

## Example: Bounded compartment (I)

Task: create a closed compartment with surrounding volume with a simple fire

- ▶ define grid spacing
  - ▶ define parameters for compartment size
  - ▶ define fire position
  - ▶ use <bounded\_room> to define mesh and walls
  - ▶ use <fire> to define a burning obstacle in the middle of the room
- hands-on example 5

## Example: Bounded compartment (II)

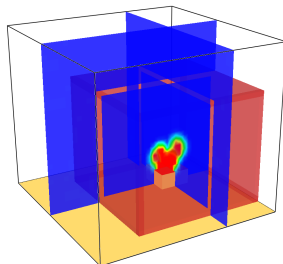
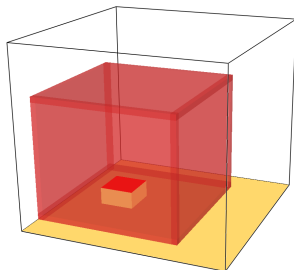
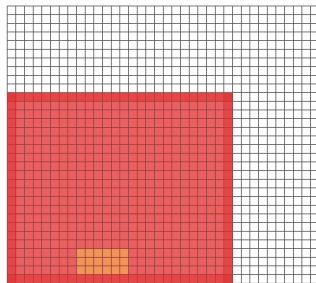
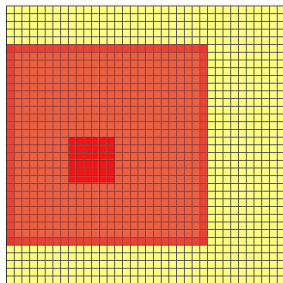
e05.xml

```

1 <fds>
2
3   <info chid="'fgg_example_05'" title="'fgg example 05'"
4     outfile="'e05.fds'" subdir="'rundir'" />
5
6   <input str="'TIME T_END=10.0'" />
7
8   <boundary x="'open'" y="'open'" zmax="'open'" />
9
10  <var delta="0.1" />
11  <var lx="2.4" ly="2.4" lz="2.0" />
12  <var fx="1.0" fy="1.0" />
13
14  <bounded_room x1="0.0" y1="0.0" z1="0.0" x2="lx" y2="ly" z2="lz"
15    wt='delta'
16    ball="1"
17    ex2="1.0" ey1="0.5" ey2="0.5" ez2="1.0" />
18
19  <fire type="'burningbox'" cx="fx" cy="fy" lz="0.0"
20    width="0.6" height="0.3" hrr="100" />
21
22  <slcf q="'TEMPERATURE', 'VELOCITY'" x="fx" y="fy" />
23
24 </fds>

```

## Example: Bounded compartment (III)





## Loops

`fdsgeogen` provides a basic support for loops to ease repetetive tasks

---

```
<var ndevc="20" dz="zmax / ndevcs" />

<loop var="i" start="0" stop="ndevc">
  <var z="zmin + dz*i" />
  [...]
</loop>
```

---

- ▶ placement of obstacles and devices
- ▶ complex obstacles (e.g. stairs) or holes (e.g. round opening)

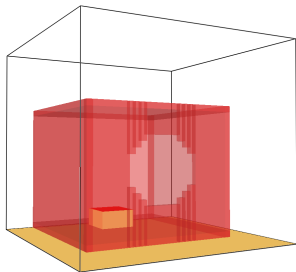
## Example: Round opening (I)

Task: create a round opening (works the same for other types)

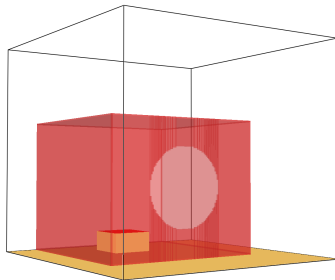
- ▶ create a hole for each cell line
- ▶ compute the width of a hole line based on opening definition
- ▶ loop over all hole lines

→ hands-on example 6

## Example: Round opening (II)



$$\Delta x = 0.10$$



$$\Delta x = 0.02$$

## Example: Round opening (III)

e06.xml

```
22 <var hole_radius="0.6" hole_z="1.0"/>
23 <var nlines='int(hole_radius / delta * 2.0)' />
24
25 <loop var='i' start='0' stop='nlines' >
26
27     <var zoff = "- hole_radius + i*delta" />
28     <var ywidth = "np.sqrt(hole_radius**2 - zoff**2)" />
29
30     <fds_hole xb='lx, lx+delta, ly/2. - ywidth, ly/2. + ywidth,
31                hole_z + zoff, hole_z + zoff + delta' />
32
33 </loop>
```

## Device output

The output of FDS devices can be captured and directly plotted. There are three different options for that – which can be combined:

- ▶ `single`, plots just the device's output
- ▶ `local:group`, combines the output of all devices with the same group for the local FDS simulation
- ▶ `global:group`, same as above but for the whole simulation ensemble

---

```
<devc id="c_T" q="..." plot="'single', 'local:T', 'local:central'" />
```

---

Note: The information for plotting is stored in the `fgg.plot` files, which can be edited also afterwards. The analysis may be started even if the full ensemble is not finished.

## Example: Flow velocities at round opening (I)

1. Define devices relative to opening position, center and  $\pm$  half radius
2. Define analysis groups
3. use `fgg_analyse` to automatically create plots, the FDS calculation has not to be finished

→ hands-on example 7

e07.xml

---

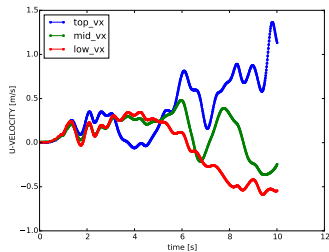
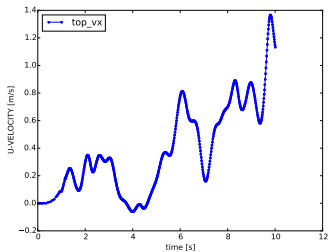
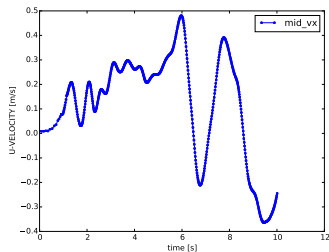
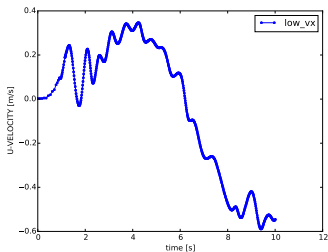
```

35 <devc id="'top_vx'" x="lx" y="ly/2." z="hole_z + 0.5*hole_radius"
36     q="'U-VELOCITY'" plot="'single', 'local:vx'" />
37
38 <devc id="'mid_vx'" x="lx" y="ly/2." z="hole_z"
39     q="'U-VELOCITY'" plot="'single', 'local:vx'" />
40
41 <devc id="'low_vx'" x="lx" y="ly/2." z="hole_z - 0.5*hole_radius"
42     q="'U-VELOCITY'" plot="'single', 'local:vx'" />

```

---

## Example: Flow velocities at round opening (II)



## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions



## Idea

In `fdsgeogen` high dimensional parameter spaces can be traversed.

For each parameter a set of values has to be defined in the `<para>` nodes.

All sets can be traversed simultaneously, i.e. they have to be of same size, or all possible parameter combinations are evaluated.

- ▶ sets to be evaluated simultaneously have to be in the same dimensional group via the `dim` attribute
- ▶ the running variable `para_id` can be used to enumerated subdirectories or other output quantities
- ▶ the parameter sets can be either explicitly stated or read out of a `csv` file

→ hands-on example 8

## Example

### e08.xml

---

```

3 <para dim="pos" var="xpos" list="1.0, 1.2, 1.4, 1.6, 1.8, 2.0" />
4 <para dim="pos" var="ypos" list="range(6)" />
5 <para dim="dia" var="diam" list="0.1, 0.2, 0.3" />
6 <para dim="hrr" var="hrr" file="input-hrr.csv"/>

```

---

### input-hrr.csv

---

```

1 4.56
2 7.89

```

---

This definition results in a total of  $6 \cdot 3 \cdot 2 = 36$  parameter combinations:

---

```

para ID: 00 -- xpos=1.000000 -- ypos=0.000000
           -- diam=0.100000 -- hrr =4.560000

para ID: 01 -- xpos=1.000000 -- ypos=0.000000
           -- diam=0.200000 -- hrr =4.560000

[...]
para ID: 34 -- xpos=2.000000 -- ypos=5.000000
           -- diam=0.200000 -- hrr =7.890000

para ID: 35 -- xpos=2.000000 -- ypos=5.000000
           -- diam=0.300000 -- hrr =7.890000

```

---

## Hole (I)

Vary the position and radius of the hole in the previous example.

- ▶ 6 different positions in  $z$  [m]: from 0.5 to 1.5
- ▶ 4 different hole radii [m]: 0.5 to 0.2
- ▶ devices will be moved accordingly to hole position and radius

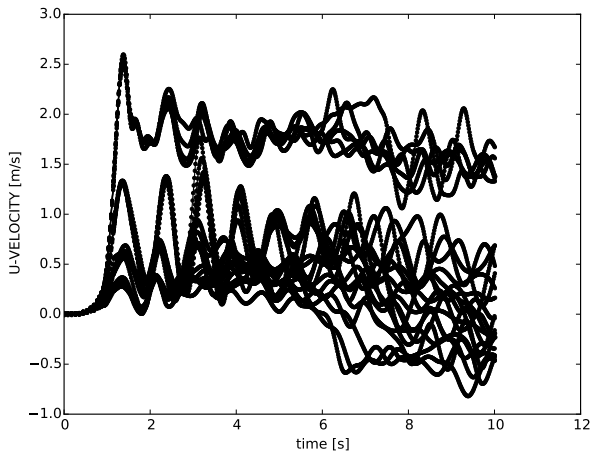
→ hands-on example 9

e09.xml

```
3 <para dim="zpos" var="hole_z" list="np.linspace(0.5, 1.5, 6)" />  
4 <para dim="radi" var="hole_radius" list="np.linspace(0.5, 0.2, 4)" />
```

## Hole (II)

In/Out flow velocity at `hole_z - 0.5 hole_radius`



## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions

## Dynamic burning surface (I)

Dynamic (w.r.t. to HRR and surface) burning surfaces can be defined in fdsgeogen via the <fire> nodes.

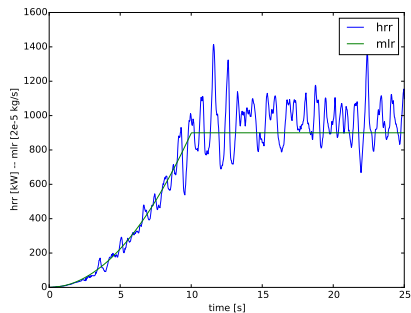
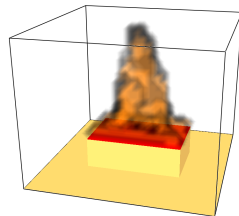
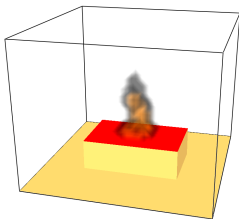
The dynamic surfaces are define as ramps for each surface element, which in total result in the chosen HRR curve. The increase in HRR is basically due to increase in burning surface.

→ hands-on example 10

e10.xml

```
31 <fire type="spread_square_box" cx='1.5' cy='1.5' lz='0.0'  
32 width_x='1.0' width_y='1.6' height='0.5'  
33 hrrmax='1000' alpha='10' />
```

## Dynamic burning surface (II)



## Other features

### Import

Import of existing FDS input files is managed by the `<input>` nodes.

They allow for selective import using either an `include` or an `exclude` attribute list.

### Batch Execution

An automated serial execution is already implemented.

Support for batch systems, like SLURM, is available.

### Domain Decomposition

Meshes can be automatically split into P sub-meshes.



## 1. fdsgeogen

### 1.1 Overview

### 1.2 XML syntax

### 1.3 Parameter spaces

### 1.4 Other features

### 1.5 Conclusions

## Conclusions

`fdsgeogen` is freely available – together with a fundamental documentation and collection of examples:

→ <https://github.com/FireDynamics/fdsgeogen>

Perfect way to get started is to get in touch with us, so that we can discuss if `fdsgeogen` is suitable for you and adopt some tools – if needed.

[firesim@fz-juelich.de](mailto:firesim@fz-juelich.de)

## 2. Python FDS data reader

### 2.1 Read data

### 2.2 Conclusions

## 2. Python FDS data reader

### 2.1 Read data

### 2.2 Conclusions

## Basic idea

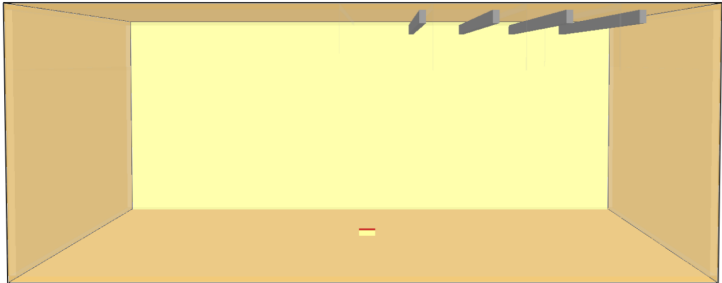
The basic idea is to have a Python module to read in various types of FDS data formats and process it in Python scripts.

Although there exists the `fds2ascii` tool to extract FDS data, the workflow with a Python module may ease your work.

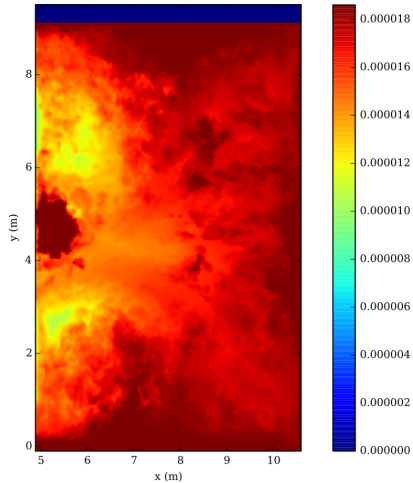
### Current implementation status:

- ▶ collection of individual scripts
- ▶ effort to make a common Python module
- ▶ is freely available and contributors are very welcome

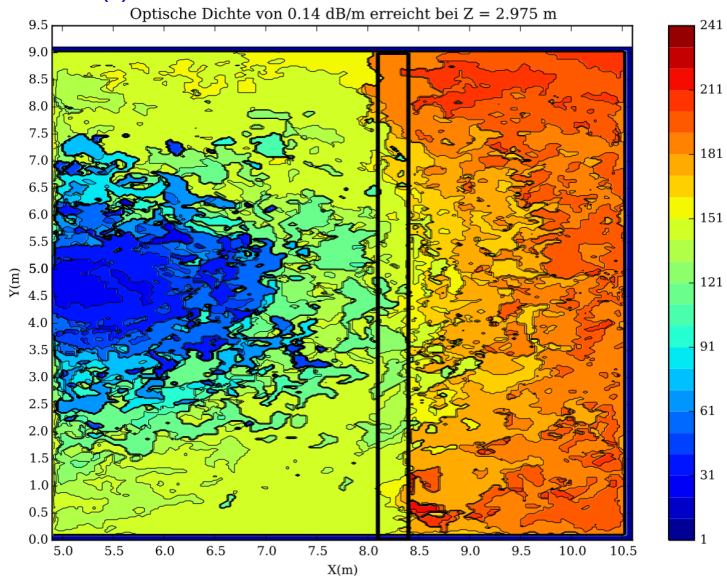
## Smoke detectors



## Optical density

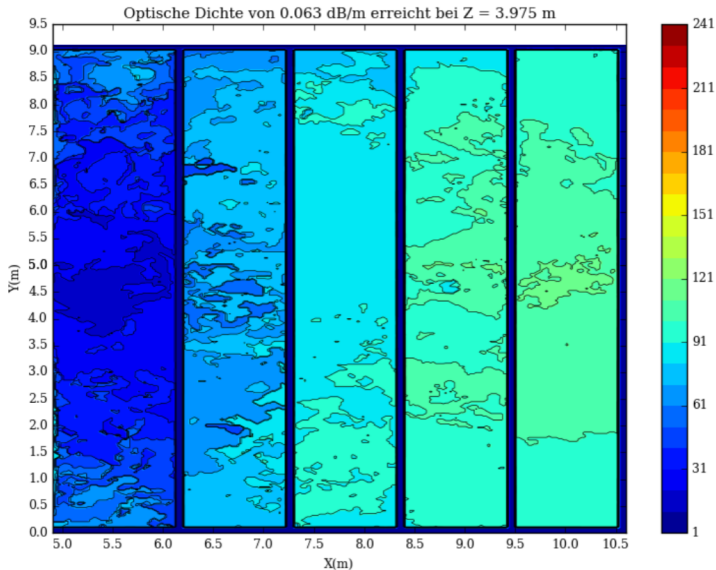


## Activation time (I)

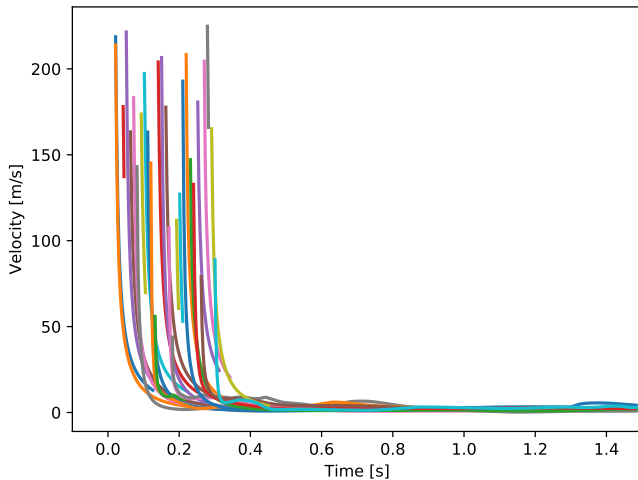




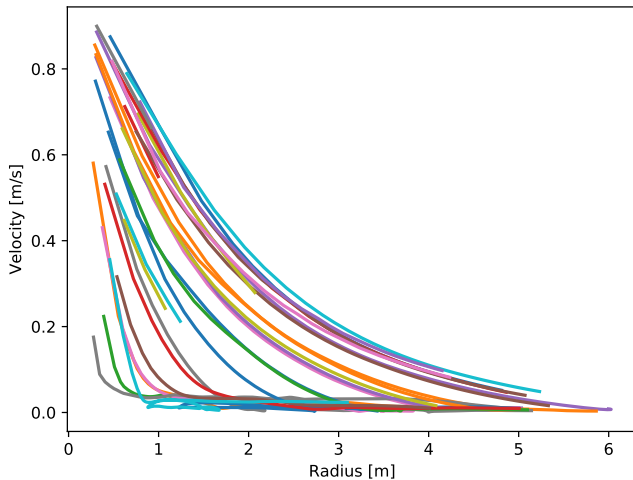
## Activation time (II)



## Particle distribution (I)



## Particle distribution (II)



## 2. Python FDS data reader

### 2.1 Read data

### 2.2 Conclusions

## Conclusions

- ▶ You can do more with data ...
- ▶ If you intend to use our reader, please contact me, as this will push (a joint) development.
- ▶ As these tools work for people, who just use them, it may not work for you out-of-the-box, please contact me and I will help you.

### 3. Multivariate Analysis

#### 3.1 Introduction

#### 3.2 Pre-processing

#### 3.3 Postprocessing

#### 3.4 Example metro station

#### 3.5 Conclusions

## 3. Multivariate Analysis

### 3.1 Introduction

### 3.2 Pre-processing

### 3.3 Postprocessing

### 3.4 Example metro station

### 3.5 Conclusions

## Why Multivariate Analysis?

Systems in fire safety engineering can easily become very complex, i.e. many independent factors with non-linear interactions.

---

geometry	environment	fire	occupants
▶ compartments	▶ temperature	▶ seat of fire	▶ number
▶ corridors	▶ wind	▶ energy release	▶ composition
▶ openings	▶ underground	▶ fire spread	▶ knowledge

---

So, what is the impact of a smoke extraction system? How will it influence the evacuation process? Will it work only for a few selected cases?

If one wants to tackle these questions with numerical modeling, a multivariate analysis may provide robust answers.



## 3. Multivariate Analysis

### 3.1 Introduction

### 3.2 Pre-processing

### 3.3 Postprocessing

### 3.4 Example metro station

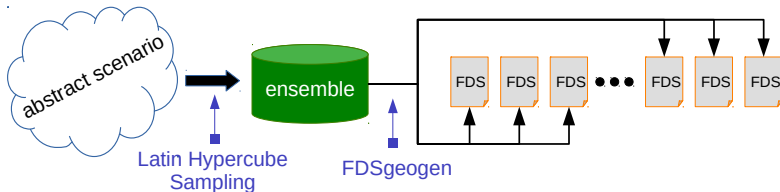
### 3.5 Conclusions

## Pre-Processing

Doing numerical simulations we have freedom to choose any parameter combination. The multi-dimensional parameter sets are to be chosen to widely represent the system.

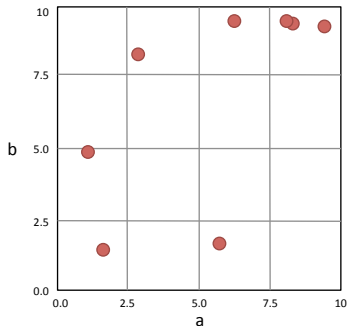
While Monte-Carlo methods randomly select parameter, orthogonal methods allow to find clever sets of parameters. A screening method may be used to reduce the number of variables.

To avoid errors and ease the work, an automated system to create input files for the CFD model should be used.



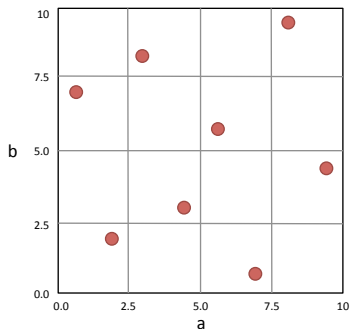
## Latin Hypercube Sampling

There exist multiple possibilities to sample the potentially high dimensional parameter space. Two possibilities:



Monte Carlo

- ▶ random sampling
- ▶ weak space filling
- ▶ easy to extend



Latin Hypercube

- ▶ prescribed pattern
- ▶ good space filling
- ▶ hard to extend

## 3. Multivariate Analysis

### 3.1 Introduction

### 3.2 Pre-processing

### 3.3 Postprocessing

### 3.4 Example metro station

### 3.5 Conclusions

## Analysis

Once the ensemble is computed, you have to analyse a huge amount of data.  
But, what is the goal?

Understand the system of interest.

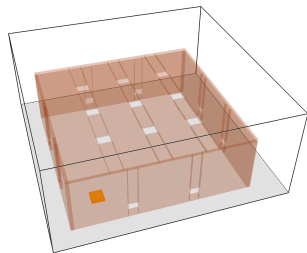
This might be quantities like:

- ▶ What will be the smoke layer height?
- ▶ What is the available safe egress time?

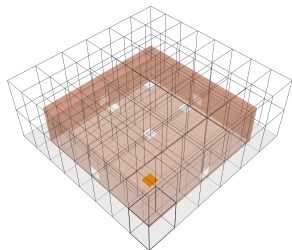
The data amount must be extremely reduced, but without losing generality!

## Data Reduction Example – Smoke Layer Height (I)

- ▶ Large compartment:  $28\text{ m} \times 28\text{ m} \times 9\text{ m}$
- ▶ Variability: geometry, fire position and properties, opening sizes and numbers
- ▶ Experimental design baseline:  
 $HHR \in [100, 10000]\text{ kW}$ ,  
 $Y_{soot} \in [0.01, 0.2]$   
 $A_{in} \in [10, 40]\text{ m}^2$ ,  $n_{in} \in \{1, 2\}$   
 $A_{out} \in [1, 20]\text{ m}^2$ ,  $n_{out} \in \{1, 2, 3\}$



- ▶ Automated domain decomposition, here into 96 meshes
- ▶ Mesh spacing: 10 cm, i.e. 18.6 million cells
- ▶ resources: about 2000 core-h per simulation



## Data Reduction Example – Smoke Layer Height (II)

How to measure the height of the smoke layer with a general approach?

1. Define a local criterion, like using a critical value for the soot density  $\rho_{s,c}$  to compare it with local soot density values  $\rho_s(x, y, z)$ :

$$h(x, y) = \min \{z | \rho_s(x, y, z) > \rho_{s,c}\}$$

2. Apply the local criterion to the full computational domain.
3. Define the global height as the most probable value.
4. Reduce temporal dimension, i.e. asymptotic behaviour.

---

Notes:

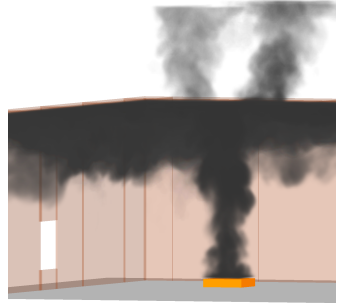
- ▶ No assumption on evaluation point needs to be made.
- ▶ Fluctuations are considered.
- ▶ More data is used.

## Data Reduction Example – Smoke Layer Height (III)

Assuming ten positions in time to look at. How big, in terms of degrees of freedom (DoF), is our data?

CFD Data: 18.6 M DoF per evaluation time

Total: 186 M DoF





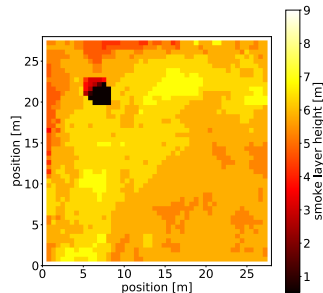
## Data Reduction Example – Smoke Layer Height (III)

Assuming ten positions in time to look at. How big, in terms of degrees of freedom (DoF), is our data?

~~CFD Data: 18.6 M DoF per evaluation time~~

Smoke layer height map:  $50^2 = 2500$  DoF per evaluation time

Total: 25 k DoF



## Data Reduction Example – Smoke Layer Height (III)

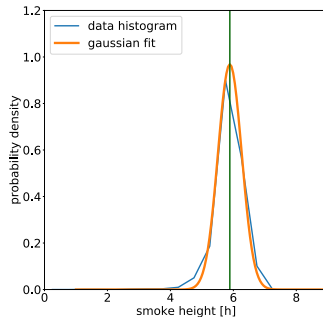
Assuming ten positions in time to look at. How big, in terms of degrees of freedom (DoF), is our data?

~~CFD Data: 18.6 M DoF per evaluation time~~

~~Smoke layer height map:  $50^2 = 2500$  DoF per evaluation time~~

Distribution function: 2 DoF per evaluation time

Total: 20 DoF



## Data Reduction Example – Smoke Layer Height (III)

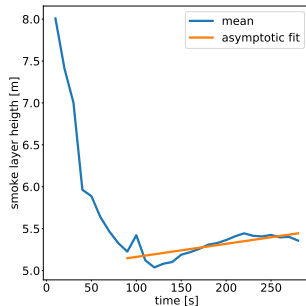
Assuming ten positions in time to look at. How big, in terms of degrees of freedom (DoF), is our data?

~~CFD Data: 18.6 M DoF per evaluation time~~

~~Smoke layer height map:  $50^2 = 2500$  DoF per evaluation time~~

~~Distribution function: 2 DoF per evaluation time~~

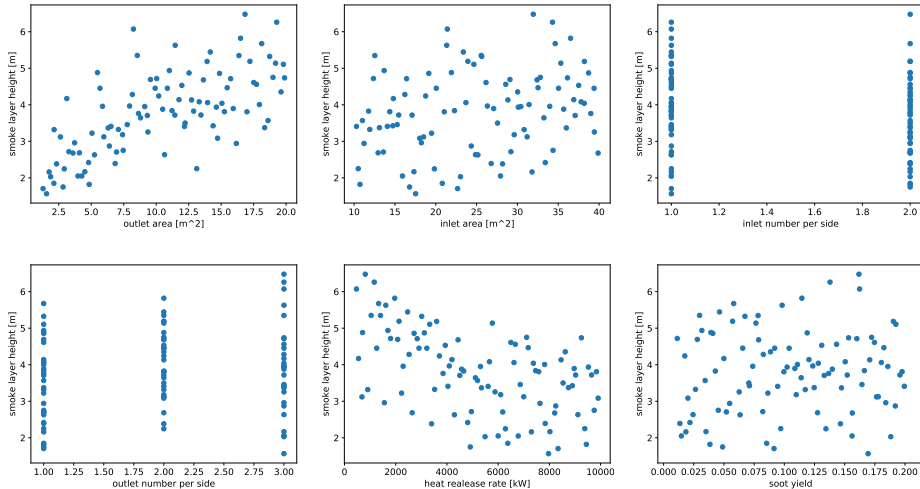
Asymptotic behaviour: 1 DoF



A single value,  
here the asymptotic mean of the smoke layer distribution,  
to describe the output.

## Analysis – Scatter Plots

Scatter plots provide a simple way to visualise data and should always be checked for consistency and plausibility.



## Analysis – Linear Model (I)

To describe the system, a linear model may provide an easy (linear) way to understand the influence of the characteristics and to utilize it in other models, e.g. risk assessment.

A linear model can cover only a part of the observational data and will in general leave an unexplained part.

The general form of a linear model with  $p$  characteristics is:

$$y = \alpha_0 + \alpha_1 x_1 + \dots + \alpha_p x_p + \sigma E$$

Where  $\alpha_i$  are the model parameters and  $\sigma E$  is the stochastic unexplained part.

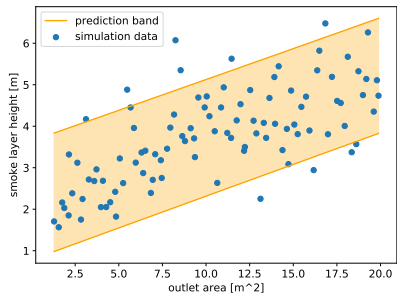
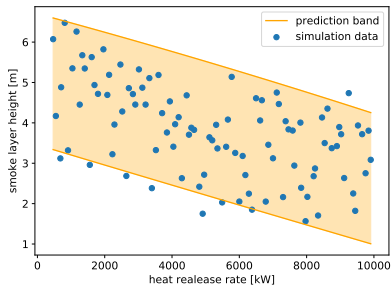
---

In our hall example, this results in the following parameters:

$$h = \left( 2.98 - \frac{2.45 \cdot 10^{-4}}{\text{kW}} \text{HRR} + \frac{0.152}{\text{m}^2} A_{out} + \frac{0.0196}{\text{m}^2} A_{in} \right) \text{m}$$

Note: Only these characteristics have a significant impact. About 85% of the data can be explained by this model.

## Analysis – Linear Model (II)



A (linear) model allows:

- ▶ to quantify the variability of the system
- ▶ to quantify the change of the output as a function of the characteristics
- ▶ to define an analytical model as input for other models

### 3. Multivariate Analysis

#### 3.1 Introduction

#### 3.2 Pre-processing

#### 3.3 Postprocessing

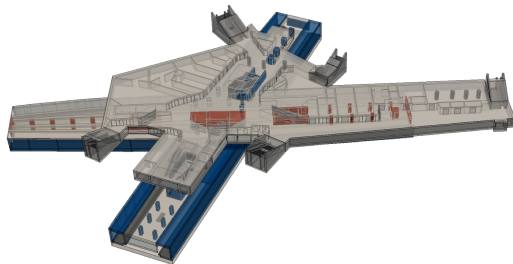
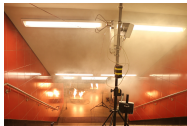
#### 3.4 Example metro station

#### 3.5 Conclusions

## Project ORPHEUS



Optimierung der Rauchableitung und Personenführung in  
U-Bahnstationen: Experimente und Simulationen



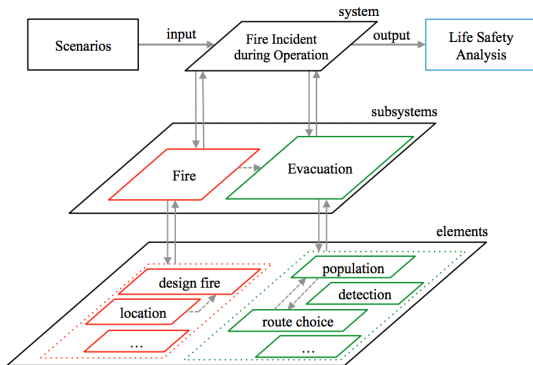
Simulation setup:

FDS: 12 meshes, 15 cm mesh, 26 M cells

JuPedSim: various routing strategies, smoke interaction



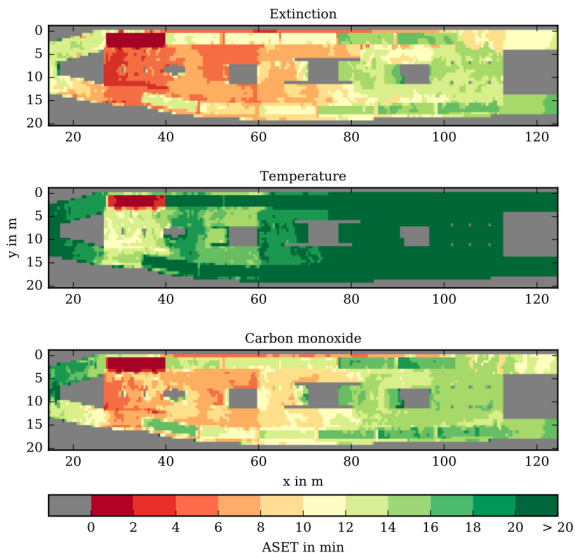
## Project ORPHEUS – Life Safety Analysis



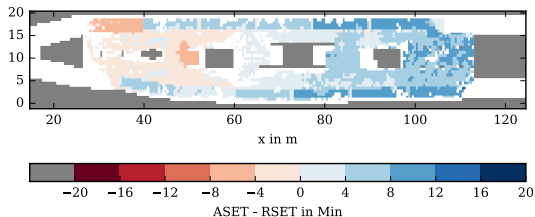
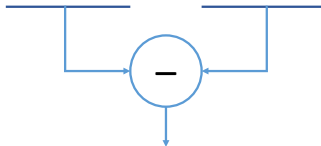
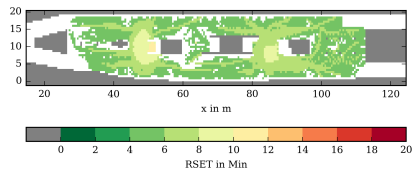
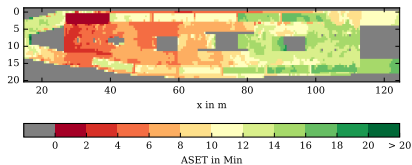
Considered scenarios:

- ▶ 108 fire scenarios (position, design, climate)
- ▶ 80 occupant scenarios (population, routing), 100 randomisation steps
- ▶ full factorial combination: **8640** scenarios

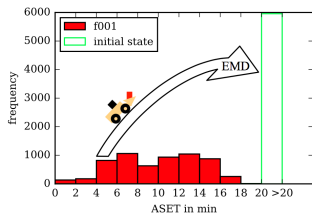
## ASET maps



## ASET/RSET difference maps

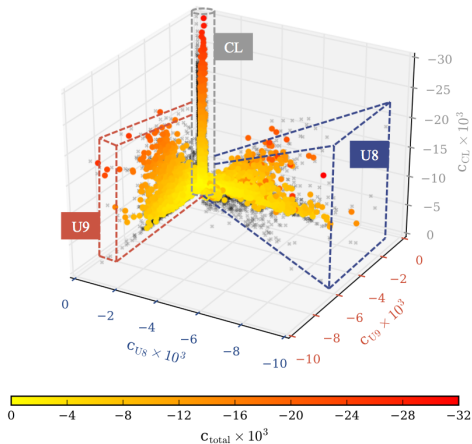


## Example Underground Station – Criticality

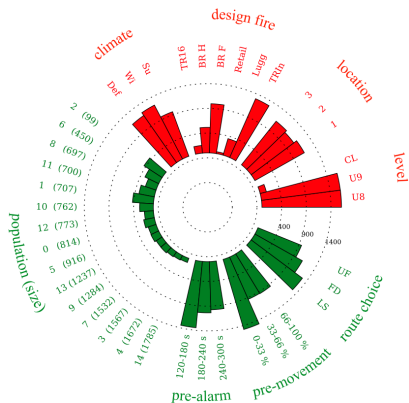


Use a metric, here the Earth Mover Distance, to characterise an ASET map.

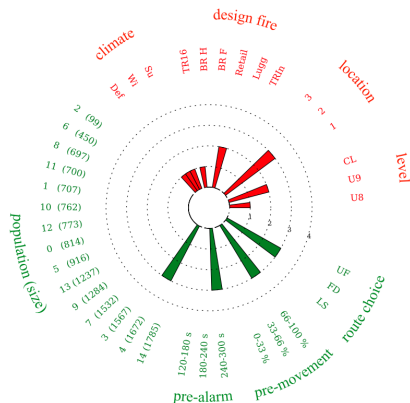
The resulting scalar, here criticality, is used to find clusters of scenarios with a similar impact on life safety.



## Example Underground Station – Cluster Analysis



Cluster A (low criticality)



Cluster J (high criticality)

### 3. Multivariate Analysis

#### 3.1 Introduction

#### 3.2 Pre-processing

#### 3.3 Postprocessing

#### 3.4 Example metro station

#### 3.5 Conclusions

## Conclusions

Multivariate analysis may provide a new viewpoint  
w.r.t. robust and quantitative evaluation  
of fire safety scenarios.

---

### Think Big:

- ▶ ensemble simulations are needed
- ▶ automation mechanisms make application practicable

### Think Small:

- ▶ big data analysis only possible via data reduction
- ▶ good reduction methods are needed