Automated FDS Input File Generation with fdsgeogen

FDS User Meeting 2015
Berlin, 12. - 13. November 2015

Lukas Arnold, Jana Boltersdorf, Andreas Meunders, Benjamin Schröder

email: l.arnold@fz-juelich.de

Jülich Supercomputing Centre, Forschungszentrum Jülich
Bergische Universität Wuppertal

Motivation

sensitivity analysis

- investigate impact of parameter multitude
- gain individual scenario independent system description
- probabilistic results possible

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 automatisation 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

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" />
```

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

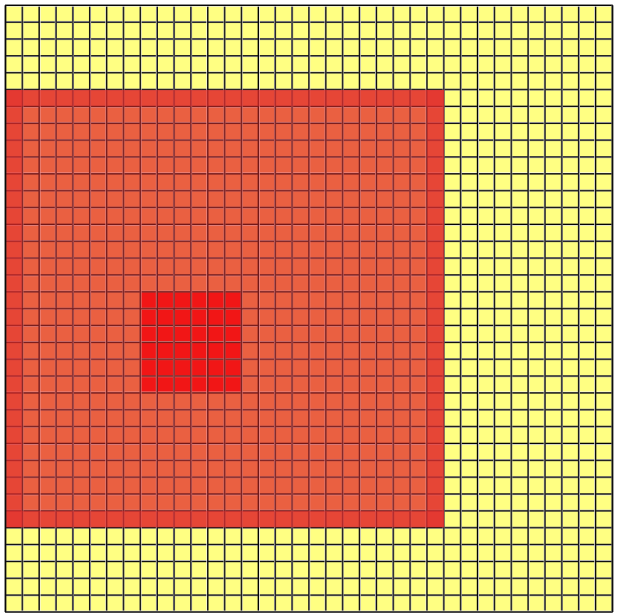
→ hands-on example 5

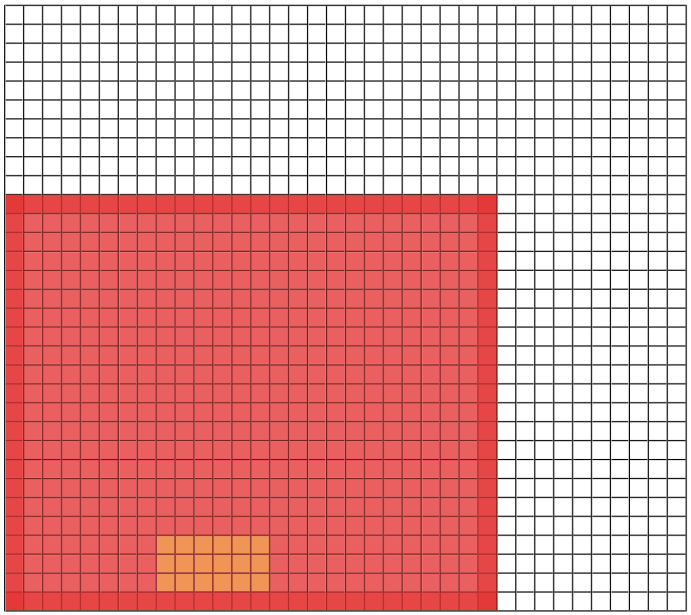
Example: Bounded Compartment (II)

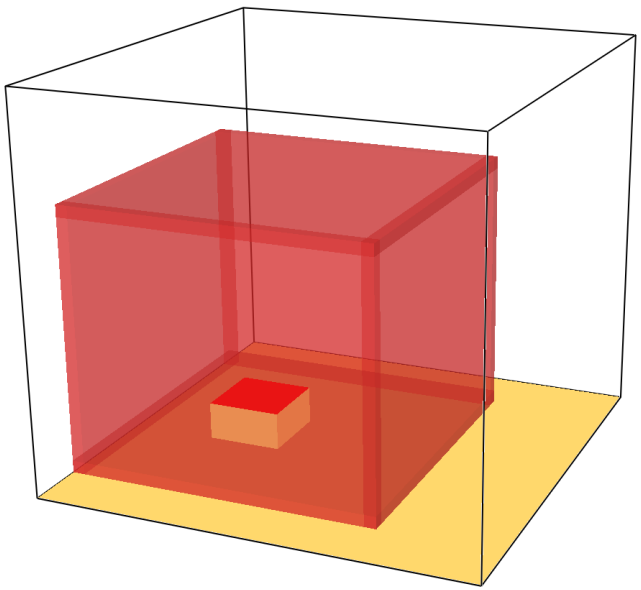
e05.xml

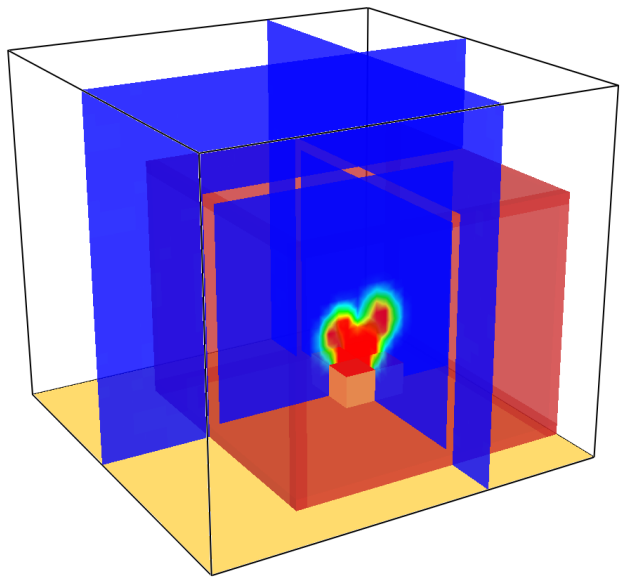
```
1  <fds>
2
3    <info chid="'fgg_example_05'" title="'fgg example 05'"
4      outfile="'e05.fds'" subdir="'rundir'" />
5
6    <input str="'TIME T_END=10.0'" />
7
8    <boundary x="'open'" y="'open'" zmax="'open'" />
9
10   <var delta="0.1" />
11   <var lx="2.4" ly="2.4" lz="2.0" />
12   <var fx="1.0" 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)

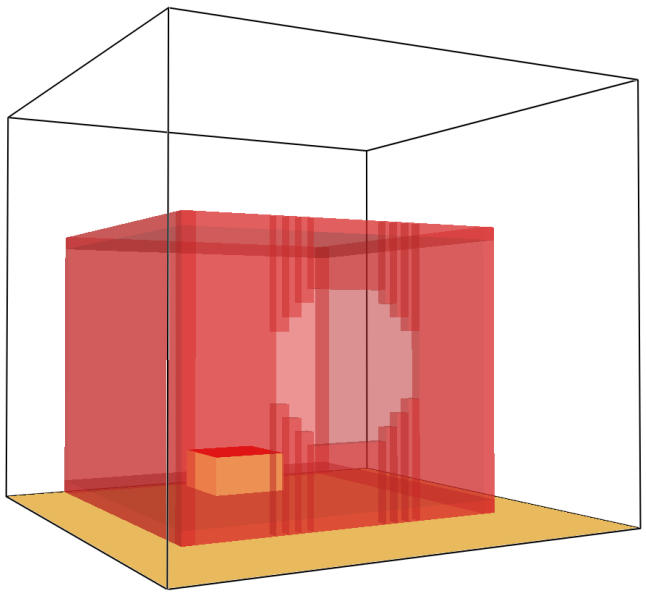
Example: Round Opening (I)

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

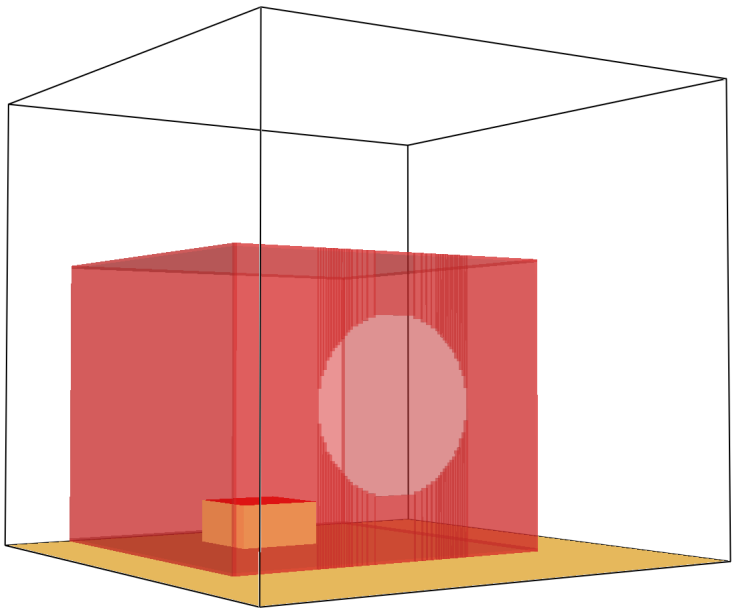
- create a hole for each cell line
- compute the width of a hole line based on opening definition
- loop over all hole lines

→ 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)

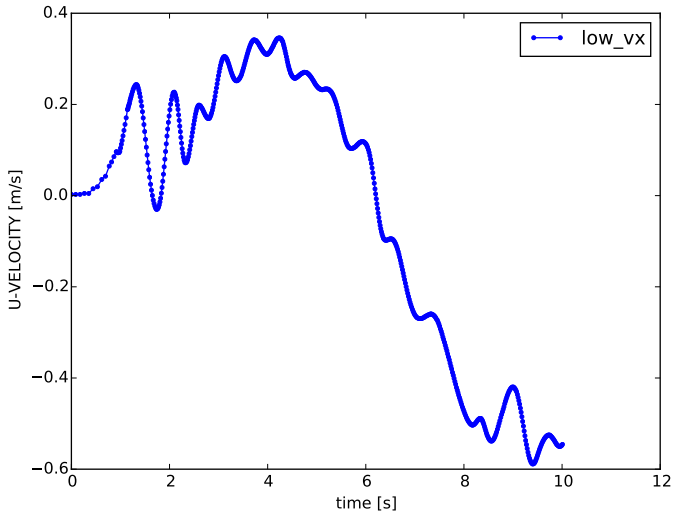
1. Define devices relative to opening position, center and $\pm$ half radius
2. Define analysis groups
3. use `fgg_analyse` to automatically create plots, the FDS calaculation has not to be finished

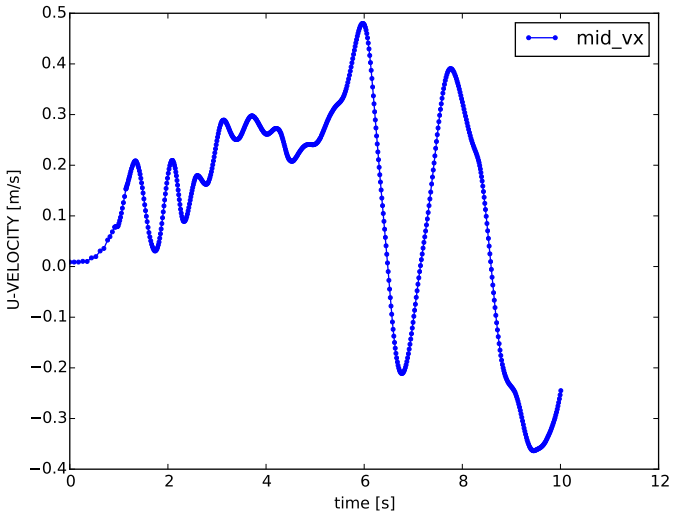
$\rightarrow$ hands-on example 7

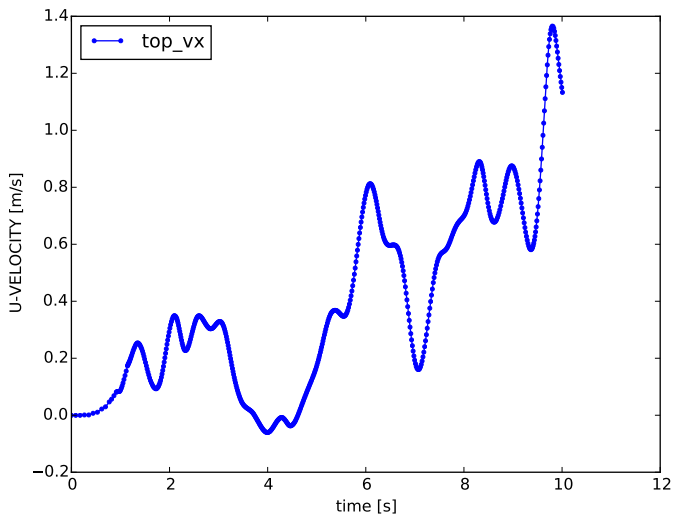
e07.xml

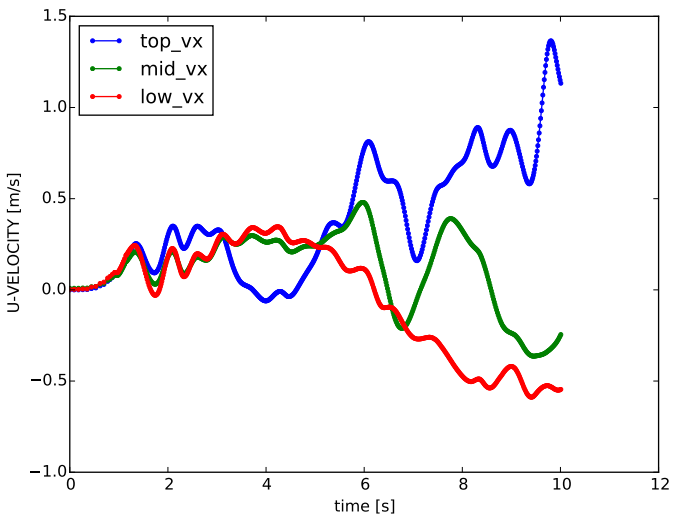
```
35   <devc id="'top_vx'" x="lx" y="ly/2." z="hole_z + 0.5*hole_radius"
36     q="'U-VELOCITY'" plot="'single', 'local:vx'" />
37
38   <devc id="'mid_vx'" x="lx" y="ly/2." z="hole_z"
39     q="'U-VELOCITY'" plot="'single', 'local:vx'" />
40
41   <devc id="'low_vx'" x="lx" y="ly/2." z="hole_z - 0.5*hole_radius"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

Example: Flow Velocities at Round Opening (II)









Parameter Space (I)

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 simultanously, i.e. they have to be of same size, or all possible parameter combinations are evaluated.

- sets to be evaluated simultanously 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

Parameter Space (II)

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
```

Parameter Space (III)

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

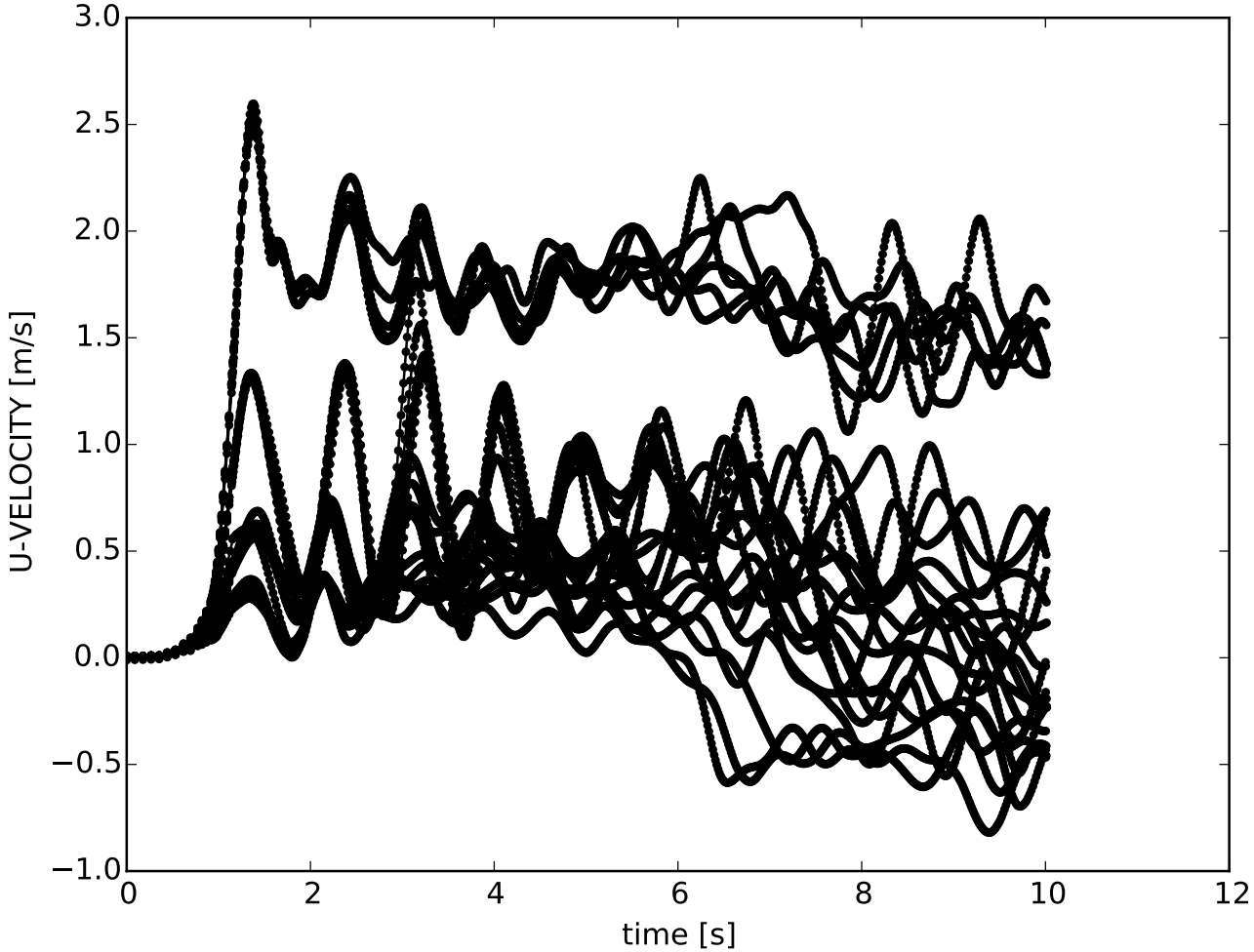
→ hands-on example 9

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)" />
```

Parameter Space (IV)

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



Dynamic Burning Surface (I)

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

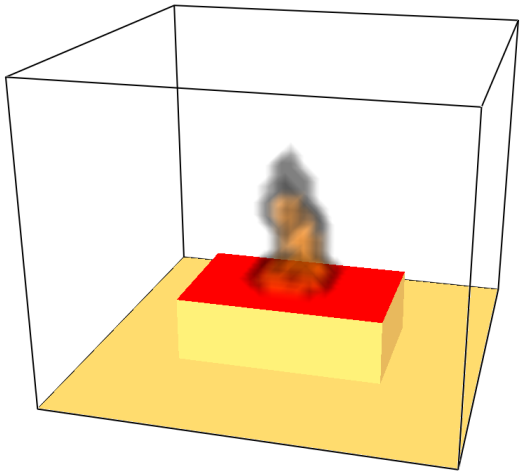
The dynamic surfaces are define as ramps for each surface element, which in total result in the chosen HRR curve. The increase in HRR is basically due to increase in burning surface.

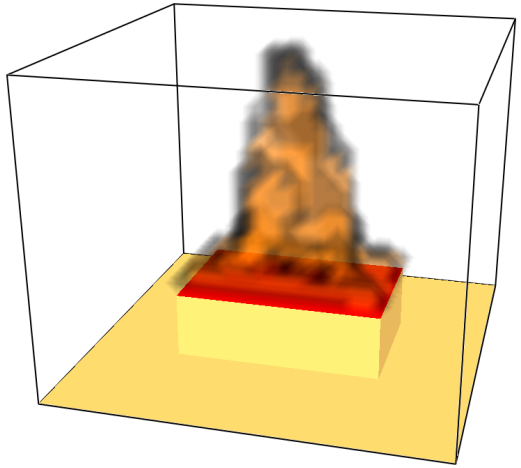
→ hands-on example 10

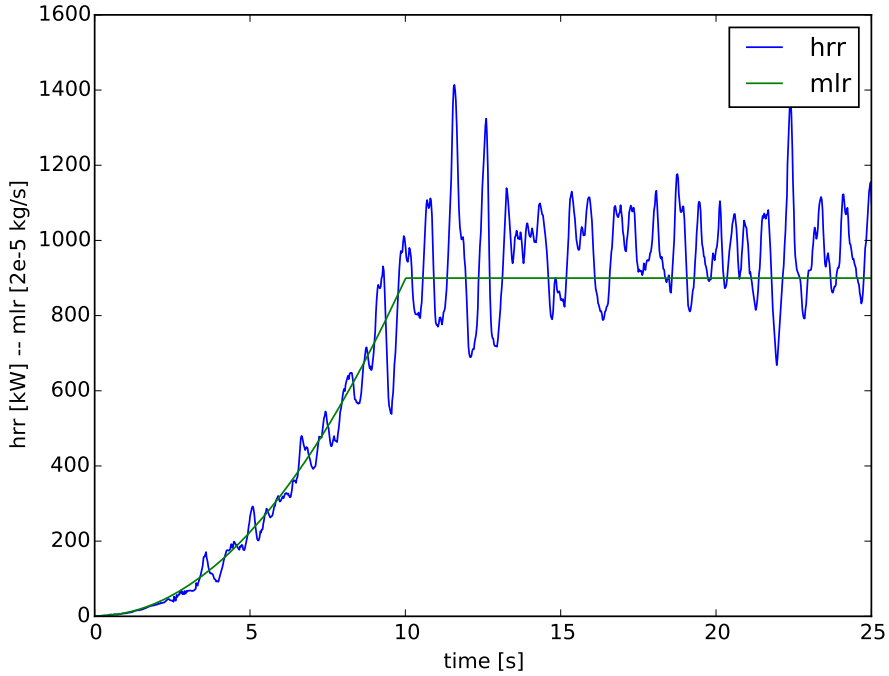
e10.xml

```
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`, are in preparation.

Domain Decomposition

Meshes can be automatically split into `P` sub-meshes.

How to get? How to get started? How to contribute?

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

→ https://cst.version.fz-juelich.de/l.arnold/fdsgeogen

Note: Repository is not open yet, next week we will move to a new `git` repository.

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`

We use a `git` repository, managed by GitLab. Although it is freely accessable, you are encouraged to get an account to be able to write issues and propose merge requests.