fds_param — FDS Parameter Study and Optimization Tool

User Guide

1. Overview

fds_param is a command-line Python tool for automating parametric studies and optimizations with FDS. It generates multiple FDS input files from a template, runs them in parallel, collects device output results, and produces a summary report.

The tool supports parameter sweeps (linear ranges), explicit value lists, parallel execution via multiprocessing, result extraction from _devc.csv files, and optional optimization using scipy.

2. Requirements

• Python 3.8+
• PyYAML (pip install pyyaml)
• SciPy — optional, for optimization mode only (pip install scipy)
• A compiled FDS binary

3. Installation

Extract the fds_param directory into Utilities/Python/ within your FDS repository. No additional installation is required beyond the Python dependencies listed above.

4. How It Works

Step 1: Create a template. Write a normal FDS input file but replace any parameter values you want to vary with double-brace tags: `{{param_name}}`

Step 2: Write a config. Create a YAML file specifying the parameter ranges, the FDS binary path, how many parallel runs, and what to extract from results.

Step 3: Run. Execute the tool and it generates all input file combinations, runs FDS for each, collects results from _devc.csv files, and outputs a summary CSV.

5. Template File Example

File: room_fire.fds.template

```
&HEAD CHID='{{chid}}', TITLE='Param study - HRRPUA={{hrrpua}}' /
&MESH IJK=20,20,20, XB=0.0,2.0,0.0,2.0,0.0,2.0 /
&TIME T_END=10.0 /
&MISC STRATIFICATION=.FALSE. /
&REAC FUEL='METHANE' /
&SURF ID='BURNER', HRRPUA={{hrrpua}}, COLOR='RED' /
&VENT XB=0.8,1.2,0.8,1.2,0.0,0.0, SURF_ID='BURNER' /
&VENT XB=0.0,0.0,0.0,2.0,0.0,2.0, SURF_ID='OPEN' /
&DEVC XYZ=1.0,1.0,1.9, QUANTITY='TEMPERATURE', ID='T_ceiling' /
&DEVC XYZ=1.0,1.0,1.0, QUANTITY='TEMPERATURE', ID='T_center' /
```

```
&TAIL /
```

The `{{chid}}` tag is automatically replaced with a unique run identifier. The `{{hrrpua}}` tag is replaced with the parameter value for each run.

6. Configuration File Example

File: param_config.yaml

```
template: room_fire.fds.template
fds_command: /path/to/fds
output_dir: ./results
parallel_runs: 2

parameters:
  hrrpua:
    type: sweep
    min: 100
    max: 300
    steps: 3

objectives:
  - device_id: T_ceiling
    quantity: max
  - device_id: T_ceiling
    quantity: final

study:
  type: parametric
```

7. Parameter Types

Type	Description	Config Example

sweep	Linear range from min to max	`type: sweep`
		`min: 100`
		`max: 300`
		`steps: 3`

list	Explicit list of values	`type: list` `values: [0.5, 1.0, 1.5]`
fixed	Single constant value	`type: fixed` `value: 10`

8. Objective Quantities

Quantity	Description
max	Maximum value of the device over the simulation
min	Minimum value of the device over the simulation
mean	Time-averaged mean value
final	Value at the last timestep
time_to_threshold	First time the device exceeds the specified threshold

9. Running the Tool

```
# Run a parametric study
python3 -m fds_param param_config.yaml

# Dry run (generate input files only)
python3 -m fds_param param_config.yaml --dry-run

# Collect results from a previous run
python3 -m fds_param param_config.yaml --collect-only
```

Note: Set PYTHONPATH to include the Utilities/Python directory if running from outside the package location.

10. Output

The tool produces:
• A subdirectory per run containing the generated .fds file and all FDS output
• A summary CSV file with columns for each parameter and each objective
• A console summary identifying the best/worst runs for each objective

Example console output:

```
Run ID          | hrrpua | T_ceiling_max | T_ceiling_final
----------------|--------|---------------|----------------
room_fire_001   | 100    | 87.6          | 72.4
room_fire_002   | 200    | 131.9         | 108.3
room_fire_003   | 300    | 175.9         | 145.7
```

11. Optimization Mode

For optimization, change the study type to 'optimization' in the config and specify which objective to optimize. The tool uses scipy.optimize to intelligently search the parameter space.

```
study:
  type: optimization
  method: Nelder-Mead
  objective: T_ceiling
  objective_quantity: time_to_threshold
  objective_threshold: 200.0
  minimize: false  # maximize time to threshold
```

12. File Structure

File	Purpose
`__init__.py`	Package initialization
`__main__.py`	CLI entry point
`config.py`	YAML config parser and parameter matrix generation
`template.py`	{{param}} substitution engine
`runner.py`	FDS execution engine (serial/parallel)
`collector.py`	_devc.csv results extraction
`optimizer.py`	scipy optimization wrapper
`report.py`	CSV and console summary output
`README.md`	Usage documentation