Rapid Indoor Diffusion Tool - User Guide

James de Lisle

July 2, 2020

Contents

1 Introduction

This document describes the usage and features of the Rapid Indoor Diffusion Tool (RIDT). RIDT is a Python implementation of two different analytical dispersion models, Eddy Diffusion and Well Mixed. It allows the calculation of concentrations and exposures based on an arbitrary set of instantaneous, infinite duration, and fixed duration source terms in a cuboid bounded system. The Well Mixed model is a set of dimensionless exponential decay equations, whereas the Eddy Diffusion model is a variant of the standard solution to the 1D diffusion equation solved with closed boundary contiditons, combined in a separable product in three dimensions. Details regarding the mathematical definitions the Well Mixed and Eddy Diffusion models can be found in Appendices B and C respectively. These models have been validated against experimental data, within their specified bounds [4].

This software was developed by Riskaware Ltd. under contract for the UK Defence Science and Technology Laboratories.

2 Installation

The following guide is for Windows. However RIDT should function on OSX, and linux operating systems.

1. Install Python 3.8.x.
   - The installer can be found here.
   - Once you have installed it to some location <path>, add both <path> and <path>/Scripts to your system path. For instructions of how to add things to your system path, please see here.
   - If you open a Command Prompt or Powershell window and type `python -V` you should see Python 3.8.x, where x is the subversion of Python 3.8 you downloaded.
   - You should also have access to the Python package manager pip. If you run the command `pip`, you should see usage instructions appear.
2. Install ridt.
   - If you have pip installed, you can simply run the command `pip install ridt` to install the latest version of ridt.

3 Basic Usage

RIDT is a command line interface tool that ingests a configuration file and outputs various data to a specified output directory. Assuming you have RIDT installed on your device as detailed in Section 2, entering the command

▷ `ridt`

into the terminal and pressing `RETURN` should yield the following console output:

```
Usage:  ridt [OPTIONS] COMMAND [ARGS]...

   The rapid indoor diffusion modelling tool (ridt).

Options:
   -help Show this message and exit.

Commands:
   csv-to-config     Merge CSV file to config JSON file.
   init              Copy a default config files current working directory.
   run               Run diffusion model.
```

We see that there are three sub commands, `csv-to-config`, `init`, and `run`. If we enter the command

```
▷ ridt run -help
```

we get the following console output:

```
Usage:  ridt run [OPTIONS] CONFIG_FILE OUTPUT_DIR

   Run diffusion model.

Options:
   -help Show this message and exit.
```

Perhaps unsurprisingly, this is the command used to run RIDT. It takes two arguments

- `CONFIG_FILE`: The path to the configuration file.
- `OUTPUT_DIR`: The path the the directory where the data generated by RIDT will be written.

A default version of the configuration file can be generated using the `ridt init` command. For example, if we run

```
▷ ridt init
```

two files will be created in the current working directory

- `config.json`: The default RIDT configuration file.
- `config.jsonc`: A commented version of the default RIDT configuration file.

The configuration file, `config.json`, is JavaScript Object Notation (JSON) format. The means it adhears to strict formatting rules. For detailed information about these rules and the contents of the configuration file please see Section 4. The file `config.jsonc` contains the same information as `config.jsonc`, in addition to comments comments giving a brief explanation of what each setting does.

We now have everything we need to run RIDT with default settings. To do so we can run the command

```
▷ ridt run config.json .
```

where we have provided the path to the configuration file which is in the current working directory, and the period at the end indicates the ouptut directory should be the current working directory. If everything runs smoothly this will result a some information about the current status of the run being output to the console. Once the run is complete the console will output `Complete.`

After the above command has successfully execute, there will be a set of new files in the current working directory.

```
   concentration_extrema.txt
   config.json
   config.jsonc
   exposure_extrema.txt
   run_summary.txt
├──domain
   ├──concentration
      ├──analysis
      └──data

   └──exposure
      ├──analysis
      └──data
├──lines
```

```
    ├──concentration
    │   ├──analysis
    │   ├──data
    │   └──plots
    └──exposure
        ├──analysis
        ├──data
        └──plots
├──planes
    ├──concentration
    │   ├──analysis
    │   ├──data
    │   └──plots
    └──exposure
        ├──analysis
        ├──data
        └──plots
└──points
    ├──concentration
    │   ├──analysis
    │   ├──data
    │   └──plots
    └──exposure
        ├──analysis
        ├──data
        └──plots
```

where we have only shown the files in the top level directory. In reality each subdirectory will contain data, images, and analysis corresponding to the run.

In the top level directory, in our case the current working directory, there are three new files of note

- `run_summary.txt`: A summary of the run configuration.
- `concentration_extrema.txt`: A list of extremal values for all evaluated concentration quantities.
- `exposure_extrema.txt`: A list of extremal values for all evaluated exposure quantities.

The file `run_summary.txt` contains a human readable run summary including some sanity check calculations. The files `concentration_extrema.txt` and `exposure_extrema.txt` contain quantites that are computed over all defined geometries. For more information about what analysis is performed and how it is presented, please see Section 7.

4 The RIDT Configuration File

4.1 JSON Format

The RIDT configuration file is a JSON format document. The basic element of a JSON document is the empty JSON *object*, `{}`. Contained within a JSON object can be one or many of the following types

- a string: "encapuslated in double quotes."
- a number: floats (1.1123) or integers (23).
- an array: [3, 1.1243, "of other types", ["including arrays"]].
- a boolean: true or false
- null: null
- another JSON object: `{}`

Each item contained in a JSON object must have a string identifier, so for example a JSON object containing one element might be

```
{
    "the_number_one": 1
}
```

Each `string: value` pair must be separated by a comma

```
{
    "the_number_one": 1,
    "dont_forget_the_comma": "never_ever"
}
```

If the comma is not present an error will occur when programs try and parse the file. This is the most common form of error you will encounter when using JSON.

The ability for JSON object to contain other JSON objects makes the format very flexible and ideal for configuration files. Those people familiar with the Python programming language will note the similarity between the Python dictionary data structure.

The standard JSON format does not permit in line comments to be added. Standard text editor linters will show any text comments as errors. Furthermore any JSON parser will throw an error if it parses anything other that standard JSON syntax. As metioned in the previous section, files with the extension `.jsonc` can contain comments of the form

```
{
    // This is an inline comment in a .jsonc file.
    // It is prepended by a double forward slash.
    "a_normal": "setting"
}
```

It is possible that you will need to install the relevant extension in your text editor if you want to see syntax highlighting in `.jsonc` files. This commented form of JSON will not work as an input file for RIDT, however it can provide useful annotations for other users, or notes for yourself.

4.2 Settings Summary

In this section we will provide detailed information about the function of each setting in the RDIT configuration file. A complete print out of the default configuration file can be found in Appendix A. We will provide snippets of the relevant section of the configuration file for each setting.

Note: any units that are not explicity defined by a setting are assumed to be SI units.

4.2.1 `eddy_diffusion`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` RIDT will evaluate the Eddy Diffusion model (Section C) with the parameters and over the domains defined in the configuration file, otherwise the model will not be evaluated.

4.2.2 `well_mixed`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` RIDT will evaluate the Well Mixed model (Section B) with the parameters and over the domains defined in the configuration file, otherwise the model will not be evaluated.

4.2.3 `compute_exposure`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` RIDT will compute, plot, and analyse the exposure.

4.2.4 `write_data_to_csv`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` RIDT will write all computed data to csv files.

4.2.5 `integration_method`

- ◇ type: string
- ◇ allowed values:
  - → "cumulativetrapezoidal"
  - → "romberg"

This string determines which method of integration to use when evaluating the eddy diffusion model. When Romberg is selected, every point in time will be integrated from zero, to a high degree of precision. When Cumulative Trapezoidal is selected, a cumulative integral is perfomed over all time points evaluated. Use Romberg when you want to evaluate a small number of time points to a high degree of precision.

4.2.6 `concentration_units`

- ◇ type: string
- ◇ allowed values:
  - → "kg.m-3"
  - → "kg.kg-1"
  - → "mg.m-3"
  - → "ppm"
  - → "ppb"
  - → "ppt"

This string determines the units in which the program ingests and outputs concentration quantities. During the execution of RIDT all calculations are done using SI units, however it can ingest and output concentration values using any of the above selections.

4.2.7 `exposure_units`

- ◇ type: string
- ◇ allowed values:
  - → "mg.min.m-3"
  - → "kg.s.m-3"

This string determines the units in which the program ingests and outputs exposure quantities. During the execution of RIDT all calculations are done using SI units, however it can ingest and output exposure values using any of the above selections.

4.2.8 `mass_units`

- ◇ type: string
- ◇ allowed values:
  - → "kg"

This string determines the units in which the program ingests and outputs mass quantities. During the execution of RIDT all calculations are done using SI units, however it can ingest and output mass values using any of the above selections. [Note: more options might be available in later versions.]

4.2.9 `time_units`

- ◇ type: string
- ◇ allowed values:
  - → "s"

This string determines the units in which the program ingests and outputs time quantities. During the execution of RIDT all calculations are done using SI units, however it can ingest and output time values using any of the above selections. [Note: more options might be available in later versions.]

4.2.10 `time_samples`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 1

This integer determines the discretisation of the temporal domain of the simulation. The times at which the models are evaluated are an evenly spaced discretisation of order `time_samples` between zero and `total_time` (Section 4.2.11), inclusive.

4.2.11 `total_time`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 1.0

This float determines the upper bound of the temporal domain of the simulation. Its units are defined by the `time_units` setting (Section 4.2.9).

4.2.12 `spatial_units`

- ◇ type: string
- ◇ allowed values:
  - → "m"

This string determines the units in which the program ingests and outputs spatial quantities. During the execution of RIDT all calculations are done using SI units, however it can ingest and output spatial values using any of the above selections. [Note: more options might be available in later versions.]

4.2.13 `dimensions`

- ◇ type: JSON Object

This object contains settings which determine the spatial bounds of the simulation.

4.2.13.1 `dimensions.x`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float defines the upper spatial bound (inclusive) along the x-axis. The lower bound is always 0.0. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.13.2 `dimensions.y`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float defines the upper spatial bound (inclusive) along the y-ayis. The lower bound is always 0.0. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.13.3 `dimensions.z`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float defines the upper spatial bound (inclusive) along the z-azis. The lower bound is always 0.0. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.14 `spatial_samples`

- ◇ type: JSON Object

This object contains settings which determine the spatial discretisation of the simulation.

4.2.14.1 `spatial_samples.x`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 2

This integer determines the discretisation of the x-axis of the simulation. The locations at which the models are evaluated are an evenly spaced discretisation of order `spatial_samples.x` between zero and `dimensions.x` (Section 4.2.13.1), inclusive.

4.2.14.2 `spatial_samples.y`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 2

This integer determines the discretisation of the y-ayis of the simulation. The locations at which the models are evaluated are an evenly spaced discretisation of order `spatial_samples.y` between zero and `dimensions.y` (Section 4.2.13.2), inclusive.

4.2.14.3 `spatial_samples.z`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 2

This integer determines the discretisation of the z-azis of the simulation. The locations at which the models are evaluated are an evenly spaced discretisation of order `spatial_samples.z` between zero and `dimensions.z` (Section 4.2.13.3), inclusive.

4.2.15 `fresh_air_change_rate_units`

- ◇ type: string
- ◇ allowed values:
  - → "m3.s-1"

This string determines the units in which the program ingests and outputs the fresh air change rate. During the execution of RIDT all calculations are done using SI units, however it can ingest and output the fresh air change rate using any of the above selections. [Note: more options might be available in later versions.]

4.2.16 `fresh_air_change_rate`

- ◇ type: float
- ◇ allowed values:
  - → >= 0.0

This setting determines the value of the fresh air change rate.

4.2.17 `physical_properties`

- ◇ type: JSON Object

This object contains settings which determine the physical quantities that are needed in the calculation of some quantities (Section 7)

4.2.17.1 `agent_molecular_weight_units`

- ◇ type: string
- ◇ allowed values:
  - → "mol.m-3"

This string determines the units in which the program ingests and outputs the agent molecular weight. During the execution of RIDT all calculations are done using SI units, however it can ingest and output the agent molecular weight using any of the above selections. [Note: more options might be available in later versions.]

4.2.17.2 `agent_molecular_weight`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float determines the value of agent molecular weight.

4.2.17.3 `pressure_units`

- ◇ type: string
- ◇ allowed values:
  - → "mol.m-3"

This string determines the units in which the program ingests and outputs the pressure. During the execution of RIDT all calculations are done using SI units, however it can ingest and output pressure values using any of the above selections. [Note: more options might be available in later versions.]

4.2.17.4 `pressure`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float determines the value of the pressure of the atmosphere.

4.2.17.5 `temperature_units`

- ◇ type: string
- ◇ allowed values:
  - → "mol.m-3"

This string determines the units in which the program ingests and outputs the temperature. During the execution of RIDT all calculations are done using SI units, however it can ingest and output temperature values using any of the above selections. [Note: more options might be available in later versions.]

4.2.17.6 `temperature`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float determines the value of the temperature of the atmosphere.

4.2.18 `modes`

- ◇ type: JSON Object

This object contains settings which define the source terms to be used in the evaluation of the models.

4.2.19 `instantaneous`

- ◇ type: JSON Object

This object contains settings which define the instantaneous source terms to be used in the evaluation of the models.

4.2.19.1 `instantaneous.sources`

- ◇ type: JSON Object

This object contains definitions of each instantaneous source (Section 4.2.19.2) to be evaluated during the run. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of sources can be defined. The only restriction is they must all have different string identifiers. For example

```
"instantaneous": {
    "sources":{
        "my_source": {...},
        "your_source": {...},
        "everyones_source": {...}
    }
}
```

If you want to remove all sources of this type, delete all entries contained within the `"sources": {}` object.

4.2.19.2 `instantaneous.sources.source`

- ◇ type: JSON Object

This object contains settings which define a single instantaneous source term to be used in the evaluation of the models. Many such sources can be defined in a single configuration file in the `instantanous.sources` setting (Section 4.2.19.1).

4.2.19.3 `instantaneous.sources.source.x`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the x-axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.19.4 `instantaneous.sources.source.y`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the y-ayis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.19.5 `instantaneous.sources.source.z`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the z-azis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.19.6 `instantaneous.sources.source.mass`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the release mass of the source. The units of this quantity are defined by the `mass_units` setting (Section 4.2.8).

4.2.19.7 `instantaneous.sources.source.time`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the release time of the source. It must lie within the temporal bounds of the simulation. The units of this quantity are defined by the `time_units` setting (Section 4.2.9).

4.2.20 `fixed_duration`

- ◇ type: JSON Object

This object contains settings which define the fixed_duration source terms to be used in the evaluation of the models.

4.2.20.1 `fixed_duration.sources`

- ◇ type: JSON Object

This object contains definitions of each fixed_duration source (Section 4.2.20.2) to be evaluated during the run. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of sources can be defined. The only restriction is they must all have different string identifiers. For example

```
"fixed_duration": {
    "sources":{
        "my_source": {...},
        "your_source": {...},
        "everyones_source": {...}
    }
}
```

If you want to remove all sources of this type, delete all entries contained within the `"sources": {}` object.

4.2.20.2 `fixed_duration.sources.source`

- ◇ type: JSON Object

This object contains settings which define a single fixed_duration source term to be used in the evaluation of the models. Many such sources can be defined in a single configuration file in the `instantanous.sources` setting (Section 4.2.20.1).

4.2.20.3 `fixed_duration.sources.source.x`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the x-axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.20.4 `fixed_duration.sources.source.y`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the y-ayis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.20.5 `fixed_duration.sources.source.z`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the z-azis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.20.6 `fixed_duration.sources.source.rate`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the release rate of the source. The units of this quantity are SI (kg.s-1).

4.2.20.7 `fixed_duration.sources.source.time`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the release time of the source. It must lie within the temporal bounds of the simulation. The units of this quantity are defined by the `time_units` setting (Section 4.2.9).

4.2.21 `infinite_duration`

- ◇ type: JSON Object

This object contains settings which define the infinite_duration source terms to be used in the evaluation of the models.

4.2.21.1 `infinite_duration.sources`

- ◇ type: JSON Object

This object contains definitions of each infinite_duration source (Section 4.2.21.2) to be evaluated during the run. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of sources can be defined. The only restriction is they must all have different string identifiers. For example

```
"infinite_duration": {
    "sources":{
        "my_source": {...},
        "your_source": {...},
        "everyones_source": {...}
    }
}
```

If you want to remove all sources of this type, delete all entries contained within the `"sources": {}` object.

4.2.21.2 `infinite_duration.sources.source`

- ◇ type: JSON Object

This object contains settings which define a single infinite_duration source term to be used in the evaluation of the models. Many such sources can be defined in a single configuration file in the `instantanous.sources` setting (Section 4.2.21.1).

4.2.21.3 `infinite_duration.sources.source.x`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the x-axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.21.4 `infinite_duration.sources.source.y`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the y-ayis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.21.5 `infinite_duration.sources.source.z`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the source along the z-azis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.21.6 `infinite_duration.sources.source.rate`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the release rate of the source. The units of this quantity are SI (kg.s-1).

4.2.21.7 `infinite_duration.sources.source.start_time`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the time the source's release starts. It must lie within the temporal bounds of the simulation. The start time must be before the end time. The units of this quantity are defined by the `time_units` setting (Section 4.2.9).

4.2.21.8 `infinite_duration.sources.source.end_time`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float determines the time the source's release ends. It must lie within the temporal bounds of the simulation. The end time must be after the start time. The units of this quantity are defined by the `time_units` setting (Section 4.2.9).

4.2.22 `thresholds`

- ◇ type: JSON Object

This object contains settings which specify any threshold values which will be used during the analysis phase of RIDT.

4.2.22.1 `thresholds.concentration`

- ◇ type: [float]
- ◇ allowed values:
  - → > 0.0

This setting is an array of float values that define the concentration thresholds that are used in the analysis phase of RIDT. The units of these quantities are defined by the `concentration_units` setting (Section 4.2.6). An example:

```
"concentration": [
    1e-10,
    1e-5,
    1e-2,
    1e-1,
    1.0
]
```

4.2.22.2 `thresholds.exposure`

- ◇ type: [float]
- ◇ allowed values:
  - → > 0.0

This setting is an array of float values that define the exposure thresholds that are used in the analysis phase of RIDT. The units of these quantities are defined by the `exposure_units` setting (Section 4.2.7). An example:

```
"exposure": [
    1e-10,
    1e-5,
    1e-2,
    1e-1,
    1.0
]
```

4.2.23 `models`

◇ type: JSON Object

This object contains settings which determine many model specific settings.

4.2.24 `models.eddy_diffusion`

◇ type: JSON Object

This object contains settings that control the way the Eddy Diffusion model is evaluated.

4.2.25 `models.eddy_diffusion.coefficient`

◇ type: JSON Object

This settings object contains settings pertaining to the selection of the means of computing the eddy diffusion coefficient.

4.2.25.1 `coefficient.calculation`

◇ type: string

◇ allowed values:

→ "TKEB"

→ "EXPLICIT"

This string determines which which method is used to calculate the eddy diffusion coefficient. If "TKEB" is selected then the coefficient is computed from the TKEB equation (Section D). If "EXPLICIT" is selected the value used in the `coefficient.value` setting (Section 4.2.25.2).

4.2.25.2 `coefficient.value`

◇ type: float

◇ allowed values:

→ ≥ 0.001

This float is the value used for the eddy diffusion coefficient if the `coefficient.calculation` setting is set to "EXPLICIT" (Section 4.2.25.1).

4.2.26 `coefficient.tkeb`

◇ type: JSON Object

This settings object contains settings pertaining to the calculation of the TKEB coefficient.

4.2.26.1 `coefficient.tkeb.bound`

◇ type: string

◇ allowed values:

→ "lower"

→ "regression"

→ "upper"

This string determines which which bound of the TKEB coefficient is used (Section D).

4.2.26.2 `coefficient.tkeb.total_air_change_rate`

- ◇ type: float
- ◇ allowed values:
  - → > 0.0

This float that specifies the value of the total air change rate to use the in calclulation of the TKEB coefficient (Section D).

4.2.26.3 `coefficient.tkeb.number_of_supply_vents`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 1

This integer determines the value of the number of supply vents to use the in calclulation of the TKEB coefficient (Section D).

4.2.27 `models.eddy_diffusion.images`

- ◇ type: JSON Object

This settings object contains settings pertaining to number of images sources used in the evaluation of the Eddy Diffusion model (Section C).

4.2.27.1 `images.mode`

- ◇ type: string
- ◇ allowed values:
  - → "auto"
  - → "manual"

This string determines the method by which images sources are added to the calculation. If "manual" is chosen the value assiged to the `images.quantity` setting is used (Section 4.2.27.2). If "auto" is chosen then the geometric variance is used to determine the corrective degree of a new image source (Section C). The geometric variance is given by

$$\exp\left[(\ln C_{\text{old}} - \ln C_{\text{new}})^2\right] \tag{1}$$

Where C_{old} and C_{old} are the tensors containing all concentrations at a given time over a given domain, before and after adding another image source.

4.2.27.2 `images.quantity`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 0

This integer determines the number if image sources to add during the evaluation of the Eddy Diffusion model. It is used if the `images.mode` setting has been set to "manual" (Section 4.2.27.1) (Section C).

4.2.28 `models.eddy_diffusion.analysis`

- ◇ type: JSON Object

This settings object contains settings pertaining to the analysis performed on the data after the models have been evaluated.

4.2.28.1 `analysis.perform_analysis`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean the analysis is performed. If `false` no analysis is performed.

4.2.28.2 `analysis.percentage_exceedance`

- ◇ type: float
- ◇ allowed values:
  - → $0.0 \leq cdot \leq 100.0$

This float defines the percentage used to calculate the time to a domain percentage exceedance of a given threshold. Please see Section 7 for more information.

4.2.28.3 `analysis.exclude_uncertain_values`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean no values within $2m$ of a source term will be included in the post run analysis. This is because within $2m$ of any source term the Eddy Diffusion model is unphysical.

4.2.29 `models.eddy_diffusion.monitor_locations`

- ◇ type: JSON Object

This settings object contains settings pertaining to the spatial domains, or geometries, over which to compute the Eddy Diffusion model.

4.2.30 `monitor_locations.evaluate`

- ◇ type: JSON Object

This settings object contains other boolean settings that switch on or off the evaluation of certain domain geometries classes.

4.2.30.1 `monitor_locations.evaluate.points`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean all point monitor locations defined in the `monitor_locations.points` setting (Section 4.2.31) will be evaluated by the model.

4.2.30.2 `monitor_locations.evaluate.lines`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean all line monitor locations defined in the `monitor_locations.lines` setting (Section 4.2.33) will be evaluated by the model.

4.2.30.3 `monitor_locations.evaluate.planes`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean all plane monitor locations defined in the `monitor_locations.planes` setting (Section 4.2.36) will be evaluated by the model.

4.2.30.4 `monitor_locations.evaluate.domain`

- ◇ type: boolean
- ◇ allowed values:
  - → true
  - → false

If `true` this boolean will mean the full domain will be evaluated by the model.

4.2.31 `monitor_locations.points`

- ◇ type: JSON Object

This settings object contains all point-like monitor locations that are to be evaluated by the eddy diffusion model. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of points can be defined. The only restriction is they must all have different string identifiers. For example

```
"monitor_locations": {
    "points":{
        "my_point": {...},
        "your_point": {...},
        "everyones_point": {...}
    }
}
```

If you want to remove all monitor locations of this type, delete all entries contained within the `"points": {}` object.

4.2.32 `monitor_locations.points.point`

- ◇ type: JSON Object

This object contains settings which define a point in space at which to evaluate the Eddy Diffusion model. Many such points can be defined in a single configuration file in the `monitor_locations.points` setting (Section 4.2.31).

4.2.32.1 `monitor_locations.points.point.x`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the point along the x-axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.32.2 `monitor_locations.points.point.y`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the point along the y-ayis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.32.3 `monitor_locations.points.point.z`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float defines the position of the point along the z-azis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.33 `monitor_locations.lines`

- ◇ type: JSON Object

This settings object contains all line-like monitor locations that are to be evaluated by the eddy diffusion model. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of lines can be defined. The only restriction is they must all have different string identifiers. For example

```
"monitor_locations": {
    "lines":{
        "my_line": {...},
        "your_line": {...},
        "everyones_line": {...}
    }
}
```

If you want to remove all monitor locations of this type, delete all entries contained within the `"lines": {}` object.

4.2.34 `monitor_locations.lines.line`

- ◇ type: JSON Object

This object contains settings which define a line in space at which to evaluate the Eddy Diffusion model. Many such lines can be defined in a single configuration file in the `monitor_locations.lines` setting (Section 4.2.33).

4.2.35 `monitor_locations.lines.line.point`

◇ type: JSON Object

This object contains settings which define the point in space through which the line passes.

4.2.35.1 `monitor_locations.lines.line.point.x`

◇ type: float

◇ allowed values:

→ ≥ 0.0

This float defines the position of this line intercept point along the x-axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.35.2 `monitor_locations.lines.line.point.y`

◇ type: float

◇ allowed values:

→ ≥ 0.0

This float defines the position of this line intercept point along the y-ayis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.35.3 `monitor_locations.lines.line.point.z`

◇ type: float

◇ allowed values:

→ ≥ 0.0

This float defines the position of this line intercept point along the z-azis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.35.4 `monitor_locations.lines.line.parallel_axis`

◇ type: string

◇ allowed values:

→ "x"

→ "y"

→ "z"

This string determines which axis lies parallel to this line.

4.2.36 monitor_locations.planes

◇ type: JSON Object

This settings object contains all plane-like monitor locations that are to be evaluated by the eddy diffusion model. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of planes can be defined. The only restriction is they must all have different string identifiers. For example

```
"monitor_locations": {
    "planes":{
        "my_plane": {...},
        "your_plane": {...},
        "everyones_plane": {...}
    }
}
```

If you want to remove all monitor locations of this type, delete all entries contained within the `"planes": {}` object.

4.2.37 monitor_locations.planes.plane

◇ type: JSON Object

This object contains settings which define a plane in space at which to evaluate the Eddy Diffusion model. Many such planes can be defined in a single configuration file in the `monitor_locations.planes` setting (Section 4.2.36).

4.2.37.1 monitor_locations.planes.plane.axis

◇ type: string

◇ allowed values:

→ "xy"

→ "yz"

→ "xz"

This string determines which axes lie parallel to this plane.

4.2.37.2 monitor_locations.planes.plane.distance

◇ type: float

◇ allowed values:

→ ≥ 0.0

This float defines the position of this plane intercept point along the perpendicular axis. It must lie within the spatial bounds of the simulation. The units of this quantity are defined by the `spatial_units` setting (Section 4.2.12).

4.2.38 monitor_locations.domain

◇ type: JSON Object

This settings object contains information about the full computational domain. It contains items which are themselves JSON objects. This setting is different from other settings in that an arbitrary number of full domains can be defined. The only restriction is they must all have different string identifiers. For example

```
"instantaneous": {
    "sources":{
        "my_domain": {...},
        "your_domain": {...},
        "everyones_domain": {...}
    }
}
```

However, given the fact that the two domains would evalualte the same information, it is currently only necessary to have a single entry in this object.

4.2.39 `monitor_locations.domain.domain`

- ◇ type: JSON Object

This object contains settings which define a plane in space at which to evaluate the Eddy Diffusion model. Many such full domain can be defined in a single configuration file in the `monitor_locations.domain` setting (Section **??**), however there is currently only need for one entry.

4.2.39.1 `monitor_locations.domain.domain.domain`

- ◇ type: bool
- ◇ allowed values:
  - → true
  - → false

This is a placeholder value and currenly serves no purpose. It must however be defined.

4.2.40 `models.eddy_diffusion.points_plots`

- ◇ type: JSON Object

This object contains settings which control in what manner plots of point-like monitor locations are produced for each run.

4.2.40.1 `points_plots.output`

- ◇ type: bool
- ◇ allowed values:
  - → true
  - → false

If this boolean is `true` then the specified line plots of the data associated with any point-like monitor locations (Section 4.2.31) are produced. This also include the Well Mixed model's outputs (which is point-like in nature). If this boolean is `false` then no plots of this type are generated.

4.2.40.2 `points_plots.scale`

- ◇ type: string
- ◇ allowed values:
  - → "linear"
  - → "logarithmic"

This string determines the type of scale to use for the dependent variable, e.g. concentration, in the plots of point-like monitor locations.

4.2.41 `models.eddy_diffusion.lines_plots`

◇ type: JSON Object

This object contains settings which control in what manner and how many graph plots of line-like monitor locations are produced for each run.

4.2.41.1 `lines_plots.output`

◇ type: bool

◇ allowed values:

→ true

→ false

If this boolean is `true` then the specified line plots of the data associated with any line-like monitor locations (Section 4.2.33) are produced. If this boolean is `false` then no plots of this type are generated.

4.2.41.2 `lines_plots.scale`

◇ type: string

◇ allowed values:

→ "linear"

→ "logarithmic"

This string determines the type of scale to use for the dependent variable, e.g. concentration, in the plots of line-like monitor locations.

4.2.41.3 `lines_plots.animate`

◇ type: bool

◇ allowed values:

→ true

→ false

This is a placeholder setting with no current function. Animations of plots as a function of time might be added in a later version.

4.2.41.4 `lines_plots.number`

◇ type: integer

◇ allowed values:

→ > 0

This integer determines the number of plots over the time domain to produce for every line-like monitor location (Section 4.2.33). It cannot be larger than the `time_samples` setting (Section 4.2.10). This will attempt to evenly space the plots over the time domain, based on the number of time samples.

4.2.42 `models.eddy_diffusion.planes_plots`

◇ type: JSON Object

This object contains settings which control in what manner and how many contour plots of plane-like monitor locations are produced for each run.

4.2.42.1 `planes_plots.output`

- ◇ type: bool
- ◇ allowed values:
  - → true
  - → false

If this boolean is `true` then the specified plane plots of the data associated with any plane-like monitor locations (Section 4.2.36) are produced. If this boolean is `false` then no plots of this type are generated.

4.2.42.2 `planes_plots.scale`

- ◇ type: string
- ◇ allowed values:
  - → "planear"
  - → "logarithmic"

This string determines the type of scale to use for contours.

4.2.42.3 `planes_plots.animate`

- ◇ type: bool
- ◇ allowed values:
  - → true
  - → false

This is a placeholder setting with no current function. Animations of plots as a function of time might be added in a later version.

4.2.42.4 `planes_plots.number`

- ◇ type: integer
- ◇ allowed values:
  - → > 0

This integer determines the number of plots over the time domain to produce for every plane-like monitor location (Section 4.2.36). It cannot be larger than the `time_samples` setting (Section 4.2.10). This will attempt to evenly space the plots over the time domain, based on the number of time samples.

4.2.42.5 `planes_plots.range`

- ◇ type: string
- ◇ allowed values:
  - → "auto"
  - → "manual"

If this string is "auto" the the maximum and minimul contours will be automatically computed from the data. This will be done over the whole time domain for a given monitor location, so all separate time correlated plots have the same contour range for better comparison. If this string is "manual" the the minimum and maximum contours defined using the values in the `planes_plots.number_of_contours` setting (Section 4.2.42.7).

4.2.42.6 `planes_plots.number_of_contours`

- ◇ type: integer
- ◇ allowed values:
  - → ≥ 2

This integer determins the number of contours used in the contour plots. The actual values for the contours a computed relative to a maximum and minimum value (either computed from the data of given explicity) and the type of scale selected.

4.2.42.7 `planes_plots.contours`

- ◇ type: JSON Object

This object contains settings which control the maximum and minimum contour values.

4.2.42.7.1 `planes_plots.contours.min`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float is the value used for the manually defined minimum contour. It must be less than the maximum value provided.

4.2.42.7.2 `planes_plots.contours.max`

- ◇ type: float
- ◇ allowed values:
  - → ≥ 0.0

This float is the value used for the manually defined maximum contour. It must be greater than the minimum value provided.

5 Batch Mode

RIDT has the ability to perform batch runs of any numerical quantity. To perfrom a batch run, we must define a quantity in the configuration file as a range. For example, take this instantaneous source

```
"source_1": {
    "x": 10.0,
    "y": 3.0,
    "z": 1.0,
    "mass": 1.0,
    "time": 0.0
}
```

We wish to evaluate the same source with varying release masses. There are two ways that ranges can be defined in a configuration file. The first is to define an explicit array of values

```
"source_1": {
    "x": 10.0,
    "y": 3.0,
    "z": 1.0,
    "mass": {
        "array": [1.0, 1.1, 1.2, 1.3]
    },
    "time": 0.0
}
```

All of the values in the array must adhere to any bounding restrictions of the quantity in question. The ordering or numerical spacing of values is not restricted. The second way we can achieve the same thing is by defining an implicit range

```
"source_1": {
    "x": 10.0,
    "y": 3.0,
    "z": 1.0,
    "mass": {
        "min": 1.0,
        "max": 1.3,
        "num": 4
    },
    "time": 0.0
}
```

where in the range values will be evenly spaced between the minimum and maximum values, inclusively.

It is possible to define more than one range in a single file. If you this is done then the cartesian product of all ranges is evaluated by RIDT. For example if three ranges are defined, one with 5 values, one with 7 values, and one with 10 values, then RIDT will evaluate the model $5 \times 7 \times 10 = 350$ times with all possible combinations of parameters.

If it is desirable that two ranges are evaluated in step, such that they are combined along the same 'axis', then they can be coupled using a "match" parameter. For example, we wish to vary the release mass of this source, but also vary the x position of the source for each mass value. To do so we provide a suplementary parameter "match" in the range definition.

```
"source_1": {
    "x": {
        "array": [10.0, 11.0, 12.0, 13.0],
        "match": "some_match_id"
    },
    "y": 3.0,
    "z": 1.0,
    "mass": {
        "min": 1.0,
        "max": 1.3,
        "num": 4,
        "match": "some_match_id"
    },
    "time": 0.0
}
```

We set the "match" id string to "some_match_id", but it doesn't matter what the provided "match" string is, as long as the two ranges have the same match string. If we run this file, RIDT will be evaluated 4 times, where each run uses the ith element of each range. You can couple an arbitrary number of

ranges with the same match parameter, and an arbitrary number of match parameters can be defined. If the length of the ranges with the same match parameter vary, then the ranges are only evaluated up to the length of the shortest range. A warning will be provided at runtime should matched ranges have different lengths.

When computing the cartesian product of the ranges to construct the computational space for the run, matched ranges form a single axis of the cartesian space. For example if we also wish to evaluate a varying y position for all values in our matched range, we can provide a range for the y parameter

```
"source_1": {
    "x": {
        "array": [10.0, 11.0, 12.0, 13.0],
        "match": "some_match_id"
    },
    "y": {
        "array": [3.0, 4.0],
    },
    "z": 1.0,
    "mass": {
        "min": 1.0,
        "max": 1.3,
        "num": 4,
        "match": "some_match_id"
    },
    "time": 0.0
}
```

In this case we have two matched ranges and a third range. If the two ranges were not matched then the computational space would be of size $4 \times 4 \times 2 = 32$. However because they are matched, the computational space is of size $4 \times 2 = 8$.

When in batch mode, the output directory will contain a number of indexed subdirectories that correspond to the position in computational space that the contained files correspond to. For example, if we defined two ranges in our configuration file both of length 2, then the computational space would be of dimension 2×2. The corresponding output directory structure would be

```
│   batch_concentration_extrema.txt
│   batch_config.json
│   batch_exposure_extrema.txt
│   batch_run_summary.txt
├──[0, 0]
│   ├──domain
│   │   ├──concentration
│   │   │   ├──analysis
│   │   │   └──data
│   │   └──exposure
│   │       ├──analysis
│   │       └──data
│   ├──lines
│   │   ├──concentration
│   │   │   ├──analysis
│   │   │   ├──data
│   │   │   └──plots
│   │   └──exposure
│   │       ├──analysis
│   │       ├──data
│   │       └──plots
│   ├──planes
```

```
    │   ├──concentration
    │   │   ├──analysis
    │   │   ├──data
    │   │   └──plots
    │   └──exposure
    │       ├──analysis
    │       ├──data
    │       └──plots
    └──points
        ├──concentration
        │   ├──analysis
        │   ├──data
        │   └──plots
        └──exposure
            ├──analysis
            ├──data
            └──plots
├──[0, 1]
├──[1, 0]
└──[1, 1]
```

where each [i, j] subdirectory containes the run config file, data, plots, and analysis for each point in the batch computational space. The file `batch_run_summary.txt` contains a summary of the batch run information, including the computational space itself.

6 csv-to-config

RIDT provides the ability to easily add large numbers or source terms and monitor locations to a configuration file. For example, if we want to add ten point monitor locations to our configuration file each with a different x position, we can use the `csv-to-config` command. If we enter the command

```
▷ ridt csv-to-config -help
```

the console will display

```
Usage:  ridt csv-to-config [OPTIONS] CONFIG_FILE CSV_FILE

   Merge CSV file to config JSON file.

Options:
   -o, -output_file PATH
   -force / -no-force
   -help              Show this message and exit.
```

As indicated it takes two arguments, the path to an existing configuration file `CONFIG_FILE`, and the path to a CSV file `CSV_FILE`. The `CONFIG_FILE` must be a valid RIDT configuration file. There is an option `-ouput` that takes a `PATH` argument that can be provided if you wish to produce a new configuration file with the new entries, leaving the old file intact. The CSV file must only contain entries of the following form

INS	float	float	float	float	float	
INF	float	float	float	float	float	
FIX	float	float	float	float	float	float
POI	float	float	float			
LIN	float	float	float	string		
PLA	string	float				

where float and string indicate the data type. Each prefix string in the first column corresponds to a different item in the configuration file

- INS: Instantaneous source. These are the ordered corresponcence between the subsequent values and the source parameters:

INS	float	float	float	float	float	
Prefix	x	y	z	mass	time	

- INF: Infinite duration source. These are the ordered corresponcence between the subsequent values and the source parameters:

INS	float	float	float	float	float	
Prefix	x	y	z	rate	time	

- FIX: Fixed duration source. These are the ordered corresponcence between the subsequent values and the source parameters:

FIX	float	float	float	float	float	float
Prefix	x	y	z	rate	star time	end time

- POI: Point monitor location. These are the ordered corresponcence between the subsequent values and the source parameters:

POI	float	float	float			
Prefix	x	y	z			

- LIN: Line monitor location. These are the ordered corresponcence between the subsequent values and the source parameters:

LIN	float	float	float	string		
Prefix	intercept-x	intercept-y	intercept-z	parallel axis		

- PLA: Plane monitor location. These are the ordered corresponcence between the subsequent values and the source parameters:

PLA	string	float				
Prefix	parallel axes	position on perpendicular axis				

For more details on the allowed values for each entry, please see the relevant sections in this document.

If we want to add ten instantaneous sources that release the same mass at 1s intervals, the CSV file will contain

INS	1.0	2.0	3.0	1.0	1.0	
INS	1.0	2.0	3.0	1.0	2.0	
INS	1.0	2.0	3.0	1.0	3.0	
INS	1.0	2.0	3.0	1.0	4.0	
INS	1.0	2.0	3.0	1.0	5.0	
INS	1.0	2.0	3.0	1.0	6.0	
INS	1.0	2.0	3.0	1.0	7.0	
INS	1.0	2.0	3.0	1.0	8.0	
INS	1.0	2.0	3.0	1.0	9.0	
INS	1.0	2.0	3.0	1.0	10.0	

When we run the command

```
▷ ridt csv-to-config config.json new.csv
```

where `new.csv` contains the information desribed in the table above, and `config.json` is our existing configuration file. Assuming the configuration file has no existing instantaneous sources defined, after executing the command above, the following instantaneous sources will be defined

```
"sources": {
        "source_0": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 1.0
        },
        "source_1": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 2.0
        },
        "source_2": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 3.0
        },
        "source_3": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 4.0
        },
        "source_4": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 5.0
        },
        "source_5": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 6.0
        },
        "source_6": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 7.0
        },
        "source_7": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 8.0
        },
```

```
        "source_8": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 9.0
        },
        "source_9": {
            "x": 1.0,
            "y": 2.0,
            "z": 3.0,
            "mass": 1.0,
            "time": 10.0
        }
    }
},
```

where the sources have been automatically numbered from 0 to 9. Any existing items with an idenfier that is the same as one that is automatically generated will be overwritten.

Any combination of different prefixes and entries can be in a single CSV file. RIDT will parse and add them to their respective section of the configuration file.

7 Run Analysis

The following analysis is perfomed on each run for every monitor location defined in the configuration file, that was marked for evaluation, for both concentration and exposure data. The data is output in CSV files in the run directory.

7.1 Maxium Values

The maximum value achieved during the course of the run. The time, location, and value are povided.

7.2 Time to Threshold Values

For all thresholds defined in the configuration file, the time during the simulation that the threshold was reached, and the location where it happened.

7.3 Time for Percent of Domain to Threshold Values

For all thresholds defined in the configuration file, the time during the simulation that the threshold was reached for a percentage of the domain. That percentage is specified in the `percentage_exceedance` setting (Section 4.2.28.2).

7.4 Maximum Percent of Domain to Exceed Threshold Values

For all thresholds defined in the configuration file the maximum percentage of the domain that exceeds the threshold, over the entire course of the run.

7.5 Extrema

For each run, a file `*_extrema.txt` is saved to the output directory that contains information about the maximal (or minimal) values over each type of monitor location for the above quantities. For example if more that one monitor point is defined, the `*_extrema.txt` will indicate which point recorded the maximum value over the entire run. For each threshold, which point reached each threshold the fastest, and so on for the other quantities. This is done separately for all types of monitor locations, including the full domain, as long as they have been marked for evaulation in the configuration file.

7.6 Batch Run Analysis

In the case of a batch run, all of the above analysis is performed and output, if specified, for each point in computational space and saved in the corresponding indexed run directory. In addition, files of the form `batch_*_extrema.txt` are saved in the parent directory containing maximal values of the analysis quantities over the entire computational space. They are of the same structure as the extrema summaries for an individual run, except they also provide the computational space index, e.g. (i, j) from our example above, to indicate which point in computational space the extremal value ocurred.

A The Full Default Configuration File

In this section we provide the full default configuration file, as created when the command `ridt init` is invoked.

```
{
    "eddy_diffusion": true,
    "well_mixed": true,

    "concentration_units": "kg.m-3",
    "exposure_units": "kg.s.m-3",
    "mass_units": "kg",

    "time_units": "s",
    "time_samples": 50,
    "total_time": 100.0,

    "spatial_units": "m",
    "dimensions": {
        "x": 50.0,
        "y": 20.0,
        "z": 3.0
    },
    "spatial_samples": {
        "x": 20,
        "y": 20,
        "z": 10
    },

    "fresh_air_change_rate_units": "m3.s-1",
    "fresh_air_change_rate": 5.0,

    "physical_properties": {
        "agent_molecular_weight_units": "mol.m-3",
        "agent_molecular_weight": 1.0,
        "pressure_units": "Pa",
        "pressure": 1.0,
        "temperature_units": "K",
        "temperature": 273.0
    },

    "modes": {
        "instantaneous": {
            "sources":{
                "source_1": {
                    "x": 10.0,
                    "y": 3.0,
                    "z": 1.0,
                    "mass": 1.0,
                    "time": 0.0
```

```
                }
            }
        },

        "infinite_duration": {
            "sources": {
                "source_1": {
                    "x": 10.0,
                    "y": 3.0,
                    "z": 1.0,
                    "rate": 0.01,
                    "time": 0.0
                }
            }
        },

        "fixed_duration": {
            "sources": {
                "source_1": {
                    "x": 10.0,
                    "y": 3.0,
                    "z": 1.0,
                    "rate": 0.01,
                    "start_time": 1.0,
                    "end_time": 0.0
                }
            }
        }
    },

    "thresholds": {
        "concentration": [
            1e-10,
            1e-5,
            1e-2,
            1e-1,
            1.0
        ],
        "exposure": [
            1e-10,
            1e-5,
            1e-2,
            1e-1,
            1.0
        ]
    },

    "models":{
        "eddy_diffusion": {
            "coefficient": {
                "calculation": "TKEB",
                "value": 0.001,
                "tkeb": {
                    "bound": "lower",
                    "total_air_change_rate": 1.0,
                    "number_of_supply_vents": 1
                }
            },
```

```
            "images": {
                "mode": "auto",
                "quantity": 3
            },
            "analysis": {
                "percentage_exceedance": 10.0,
                "exclude_uncertain_values": true
            },
            "monitor_locations": {
                "evaluate": {
                    "points": true,
                    "lines": true,
                    "planes": true,
                    "domain": true
                },
                "points": {
                    "point_1": {
                        "x": 10.0,
                        "y": 5.0,
                        "z": 1.0
                    }
                },
                "lines": {
                    "line_1": {
                        "point": {
                            "x": 10.0,
                            "y": 5.0,
                            "z": 1.0
                        },
                        "parallel_axis": "x"
                    }
                },
                "planes": {
                    "plane_1": {
                        "axis": "xy",
                        "distance": 1.0
                    }
                },
                "domain": {
                    "domain": true
                }
            },
            "points_plots": {
                "output": true,
                "scale": "logarithmic"
            },
            "lines_plots": {
                "output": true,
                "scale": "logarithmic",
                "animate": true,
                "number": 3
            },
            "planes_plots": {
                "output": true,
                "animate": true,
                "number": 10,
                "number_of_contours": 10,
                "range": "auto",
```

```
                    "scale": "logarithmic",
                    "contours": {
                        "min": 0.0,
                        "max": 1.5
                    }
                }
            }
        }
    }
}
```

B The Well Mixed Model

B.1 Instantaneous release

Equation for concentration, $C(t)$ [kg m^{-3}], in a well-mixed room from an instantaneous release of material with a mass, M [kg].

$$C(t) = \frac{M}{V} \exp\left(-\lambda_f\, t\right), \tag{2}$$

where V [m] is the room volume, λ_f [h^{-1}] is the fresh air change rate and t [s] is the time. λ_f is given $\frac{Q_f}{V}$, where Q_f [m^3 s^{-1}] is the flow rate of fresh air into the room.

B.2 Constant and continuous release

Equation for concentration in a well-mixed from a constant and continuous release

$$C(t) = \frac{S}{Q_f}\left[1 - \exp\left(-\lambda_f\, t\right)\right], \tag{3}$$

where S [kg s^{-1}] is the release rate.

B.3 Constant release rate with finite duration

Equation for concentration in a well-mixed from a constant release rate with a finite duration.

$$C(t) = \begin{cases} C(t) = \frac{S}{Q_f}\left[1 - \exp\left(-\lambda_f\, t\right)\right] & (t \leq t_{end}) \\ C(t) = C_{t_{end}} \exp\left[-\lambda_f\, (t - t_{end})\right] & (t > t_{end}), \end{cases} \tag{4}$$

where t_{end} [s] is the duration of the release, when the release starts at $t = 0$.

C Eddy Diffusion Model

C.1 Instantaneous release

The equation for concentration resulting from an instantaneous point release in a cuboid bounded space [1].

$$C(t) = \frac{M\, \exp(-\lambda_f\, t)}{8(\pi\, D_e\, t)^{\frac{3}{2}}} r_x(t) r_y(t) r_z(t) \tag{5}$$

where D_e [m^2 s^{-1}] is the eddy diffusion coefficient, the terms $r_x(t)$, $r_y(t)$ and $r_z(t)$ are given by the following:

$$r_x(t) = \sum_{n=-\infty}^{\infty} \left[\exp\left(\frac{-(x + 2nL - x_0)^2}{4D_e\, t}\right) + \exp\left(\frac{-(x + 2nL + x_0)^2}{4D_e\, t}\right)\right] \tag{6}$$

$$r_y(t) = \sum_{n=-\infty}^{\infty} \left[\exp\left(\frac{-(y + 2nW - y_0)^2}{4D_e\, t}\right) + \exp\left(\frac{-(y + 2nW + y_0)^2}{4D_e\, t}\right)\right] \tag{7}$$

$$r_z(t) = \sum_{n=-\infty}^{\infty} \left[\exp\left(\frac{-(z + 2nH - z_0)^2}{4D_e\, t} \right) + \exp\left(\frac{-(z + 2nH + z_0)^2}{4D_e\, t} \right) \right] \tag{8}$$

Where n is the number of image sources, x [m], y [m] and z [m] are the coordinates of the location which is being interrogated. x_0 [m], y_0 [m] and z_0 [m] are the coordinates of the source and L [m], L [m] and H [m] are the length, width and height of the room.

C.2 Constant and continuous release

The equation for concentration, $C_{cont}(t)$ [kg m^{-3}], resulting in a cuboid bounded space [2].

$$C_{cont}(t) = \int_0^t \frac{S\, \exp(-\lambda_f\, t)}{8(\pi\, D_e\, t)^{\frac{3}{2}}} r_x(t) r_y(t) r_z(t)\, dt \tag{9}$$

C.3 Constant release rate with finite duration

The equation for concentration, $C_{finite}(t)$ [kg m^{-3}], resulting from a finite duration point release in a cuboid bounded space.

$$C_{finite}(t) = \begin{cases} C_{cont}(t) & (t \le t_{end}) \\ C_{cont}(t) - C_{cont}(t - t_{end}) & (t > t_{end}) \end{cases} \tag{10}$$

D Equations to calculate the eddy diffusion coefficient

The turbulent kinetic energy balance (TKEB) relationship to calculate D_e is given below. There are three equations as there is some uncertainty over what what value D_e should take. The upper prediction interval (PI) equation gives the upper value for D_e, the lower PI equation gives the lower value and the regression equation gives the mid-point value.

$$D_e(\text{upper PI}) = 0.827 \frac{Q}{\sqrt[3]{V\, N^2}} + 0.0565\, \frac{\text{m}^2}{\text{s}}, \tag{11}$$

$$D_e(\text{regression}) = 0.824 \frac{Q}{\sqrt[3]{V\, N^2}}, \tag{12}$$

$$D_e(\text{lower PI}) = 0.822 \frac{Q}{\sqrt[3]{V\, N^2}} - 0.0565\, \frac{\text{m}^2}{\text{s}}, \tag{13}$$

where Q [m^3 s^{-1}] is the total supply flow rate and N is the number of supply vents. A lower bound for D_e should be applied when using these equations, which is 1×10^{-3} m^2 s^{-1}.

E Time to Well Mixed

The equation used to calculate the time taken for the room to become well mixed, in the case of the Eddy Diffusion model is [3]

$$t_{\text{wm}} = \frac{\sigma_c}{\mu_c} \le 0.1 \tag{14}$$

where σ_c is the standard deviation of the conentration over the domain, and μ_c is the spatially averaged conentration over the domain.

F Upper Exposure Limit

In the case of an instantaneous or finite duration release, an upper limit of the exposure due to each source can be computed as follows [5]

$$E_{\text{upper}} = \frac{M}{Q_f} \tag{15}$$

where M is the total mass released over the lifetime of the source and Q_f is the fresh air change rate.

References

[1] Drivas *et al.*, *Modeling indoor air exposure from short-term point source releases*, Indoor Air **6** 271-277 (1996)

[2] Cheng *et al.*, *Modeling exposure close to air pollution sources in naturally ventilated residences: Association of turbulent diffusion coffcient with air change rate*, Environ. Sci. Technol. **45** 9 4016-4022 (2011)

[3] A.C. Drescher *et al.*, *Mixing of a point-source indoor pollutant by forced convection*, Indoor Air **5** 204-214 (1995).

[4] T. Foat *et al. A relationship for the diffusion coefficient in eddy diffusion based indoor dispersion modelling*, Building and Environment **169** 106591 (2020)

[5] Parker, S., *et al.*, *Visual assessment of contaminant impacts in multizone buildings*. Building and Environment, **102**, 39-53 (2016).