Transport Temperature Model

Documentation

Sammy Breen

Contents

1 Documentation

1.1 Input file

A transport simulation can be executed in a folder that contains a transport.json file. This is the main input file and includes all information about the start time of the transport, the transport route and the cargo.

The first entry is the **type** of transport:

```
1 "type": "container",
```

The model contains four transport types

- container
- container40
- carrier
- car

The type **container** models transports in a 20-foot-long container, **container40** a 40-foot-long container, **carrier** a truck carrier and **car** is for car transports. The transport types are defined at the start of the file case.py.

The key **start** sets the start date and time of the transport:

```
2 "start": "2019-03-02 05:23:00",
```

The date and time are given in ISO 8601 format **YYYY-MM-DD hh:mm:ss**.

The temperature of the cargo and the air inside the carrier at the start of the transport are given with the key **initial_temperature**. With **arrival_temperature** one can set the ambient temperature at the destination of the transport. This is needed for simulation of heat exchange after the transport in stationary surroundings.

```
3 "initial_temperature": 20,
4 "arrival_temperature": 20,
```

The model contains two variants to define transport routes. The first option is to use the rounting service of the Institute of Automotive Technology. This service is based on the routing engine by OpenStreetMap (OSM). To use a route from this service, only the start and end coordinates of the route are needed. Coordinates are given in the format $[\mathrm{lat}, \mathrm{lon}]$.

```
"route": {
    "type": "FTM",
    "start_coordinates": [
        52.51868,
        13.370865
    ],
    "end_coordinates": [
        48.265588,
        11.671388
    ]
},
```

This method only works for routes on streets and in europe.

The second option is to use gps data of a transport. For this the name of the file, that contains the gps data, has to be set. The file needs to be in the same directory as the transport.json file. The model supports gpx and csv files. The optional keys **timezone**, **trimstart**, **trimend** can be used to modify the gps data. **timezone** is used to specify the timezone of the time readings, if they are not in UTC. **trimstart** and **trimend** can trim the data at the start and end. This is useful for data files, that start before the actual transport or contain readings after the transport.

```
"route": {
    "filename": "gpsdata.gpx",
    "timezone": "+01:00",
    "trimstart": "02:19:00",
    "trimend": "00:00:00"
},
```

The key **cargo** specifies a list with all cargo entries. Each entry has a **type**, **templateSTL**, **position**, **orientation** and **freight**. The model distinguishes between the two cargo types “Pallet” and “Car”. “Pallet” is used for all packaged types of cargo. “Car” can only be used with car transports and uses the car model.

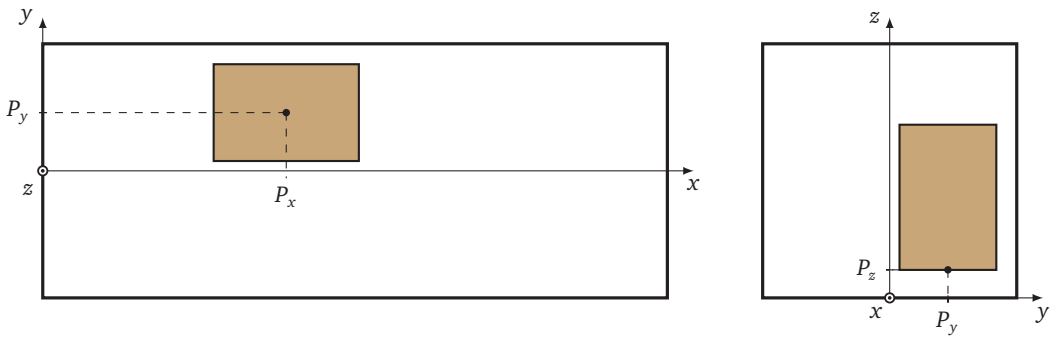

Figure 1.1: Positioning of cargo in carrier

For cargo of th type “Pallet” the position and orientation has to be set. Figure 1.1 shows how the position system works. The orientation is set as rotations around the axes in degrees. Should the cargo be a car, the keys **position** and **orientation** are not needed.
Under **freight** the information of one item of the packaged goods is given. These inputs are used to calculate the thermal properties of the cargo The needed inputs are **type**, **dimensions**, **weight**, **thermalcapacity**, **thermalconductivity**. The dimensions of the freight determines the

orientation in the package. The thermal conductivity is set as a vector because of the anisotropic behaviour of battery cells. The dimensions and the thermal conductivity are rotated according to **orientation**.

```
"cargo": [
    {
        "type": "Pallet",
        "templateSTL": "pallet3x4.stl",
        "position": [
            1.3601,
            -0.54,
            0.144
        ],
        "orientation": [
            0,
            0,
            90
        ],
        "freight": {
            "type": "cells",
            "dimensions": [
                0.173,
                0.125,
                0.045
            ],
            "weight": 2.06,
            "thermalcapacity": 1243,
            "thermalconductivity": [
                0.48,
                0.48,
                21.0
            ]
        }
    }
]
```

1.2 Terminal Commands

The program uses a series of commands to control the simulation. The simulation can be started with the simple command

```
$ ttm
```

If the current directory is not the directory where the simulation shall be executed, one can use the command

```
$ ttm --transport path_to_directory
```

The default configuration uses parallelisation with four local CPU cores. If desired, a custom core count can be set with the command

```
$ ttm --cpucores core_count
```

With the flag savetimes all timedirectories of the OpenFOAM simulation are saved. Otherwise only the last two times are saved.

```
$ ttm --savetimes
```

The parallelisation deconstructs the case in multiple regions. If the simulation results should be visualised, the case needs to be reconstruct. The flag reconstruct does that.

```
$ ttm --reconstruct
```

If the simulation case needs to be transferd to another device, it can be compressed with the flag pack.

```
$ ttm --pack
```

The model contains a method to read the temperature from a specific location in the mesh. This probing needs the name of the region and the coordinates of the location.

```
$ ttm --probe regionname '(x y z)'
```

Probe locations can also be set in a csv file with the columns region, x, y and z named probe_locations.csv. This file has to be in the transport directory.

```
$ ttm --probe file
```

The heat exchange after the transport can be simulated with the command

```
$ ttm --arrival
```

The clone flag overwrites the existing transport case and clones a new template case.

```
$ ttm --clone
```

New weather data can be gathered with the weather flag.

```
$ ttm --weather
```