nestli

Release 0.1.0

Aaron Bojarski

Sep 13, 2022

CONTENTS:

CHAPTER

ONE

NESTLI README

nestli is the little sister of the NEST demonstrator at Empa. It is co-simulation environment for benchmarking the performance of BACS (building automation and control systems). In its core, nestli is a calibrated EnergyPlus model of the UMAR living lab. see: https://www.empa.ch/web/nest/urban-mining

The calibrated EnergyPlus model is wrapped into an FMU (fucntional mock-up unit) using the EnergyPlusToFMU tool. see: https://simulationresearch.lbl.gov/fmu/EnergyPlus/export/

The model is calibrated on measurements that are collected at 1-minute intervals, and thus runs at the same temporal resolution. The HVAC system can be controlled by overriding the setpoint temperature in each room. It is also possible to evaluate the robustness of the controller by manipulating weather conditions and building operation.

1.1 Installation

The installation consists of 3 Steps.

1. EnergyPlus
2. Python
3. install the nestli package

Further installation informations are in the pdf documentation.

1.2 Usage

A simple example of how to use the package is given in the **example** folder. Just run the file nestli_example_run.py where the config example_config.yml specifies all the parameters.

CHAPTER

TWO

INSTALLATION

2.1 Local python Installation

2.1.1 EnergyPlus

Make sure you get the correct version.

1. Install Energyplus 9.3.0
2. Add it to your PATH

2.1.2 Python

1. Install python version 3.8 or higher.
2. Create a virtual environment: `python -m venv path/to/env/nestli-venv`
3. Activate the environment.

2.1.3 nestli package

1. Clone the repository onto you machine
2. cd to the directory where you cloned it into
3. Install the package with: `pip install -e .` This will install the nestli package according to setup.py.

2.2 Docker

To run the simulation in Docker you just need to have Docker installed. Then clone the repository and navigate to it.

You create a Docker image with:

```
docker build . -t nestli
```

CHAPTER

THREE

USAGE

3.1 Overview

The simulation is handled, initialized and started my the Manager class.

To run a simulation, you need to start a Manager instance with a config file. The config file specifies all simulation parameters.

3.1.1 Script

The following python script is all that is needed for a simulation.

```
example_manager = Manager("example_config.yml")
example_manager.build_simulation()
example_manager.run()
```

3.1.2 Config file

The config file is a yaml file in which the simulation parameters are set. It first specifies some general parameters.

```
OUTPUT_FOLDER_PATH: ""
START: '2022-05-12'
DURATION: 864000
RESOLUTION: 60
```

For an EnergyPlus model the DURATION has to be a multiple of 86400 and the RESOLUTION must be the same as the model resolution.

After that all the simulators are specified. A PATH can either be absolute or relative to the config file folder path or if it is in the nestli package you can see the example below.

```
# The Simulators specify all the models in the simulation. Currently the following␣
↪simulators are available:
#   FMU:  This will create a simulator from an FMU.
#         PATH specifies the folder where the FMU is contained and NAME the filename.
#   TABULAR_DATA: This will create a simulator from DATA in tabular form. The supported␣
↪format is hdf5.
#         PATH specifies the folder where the .h5 files are contained. The file name␣
↪must correspond with the variable name of the data it containes.
#   NAN_PLACEHOLDER: This simulator will create a NaN value to all connected entities.
```

(continues on next page)

(continued from previous page)

```
#         PATH specifies a file which containes all the variables you plan to connect to␣
↪it.
#   COLLECTOR: This is a simple Data collector. It will save the data of all signals you␣
↪connect to it and write them to a file after the simulation.
#         No additional information is neccessary.

SIMULATORS:
    SIM1:
        NAME: "UMAR"
        TYPE: "FMU"
        PATH: "$nestli$./simulators/ressources/fmu"
        NUMBER_OF_MODELS: 5
```

Last all the connections between the simulators are defined. For each simulator you specify all the outgoing signals.

```
MAPPINGS:
    SIGNALS_OUTGOING_FROM_UMAR:
        FROM: SOURCE
        MAPPINGS:
            # A simple mapping
            MAPPING1:
                TO: DESTINATION
                VARIABLES:
                    SOURCE_VARIABLE_NAME_1: DESTINATION_VARIABLE_NAME_1
                    SOURCE_VARIABLE_NAME_2: DESTINATION_VARIABLE_NAME_2
            # A mapping with circular dependency and multiple models and thus an index␣
↪has to be specified
            MAPPING2:
                TO: DESTINATION
                INDEX: 0
                CIRCULAR_DEPENDENDY: TRUE
                INITIAL_VALUES:
                    SOURCE_VARIABLE_NAME: 24.5669
                VARIABLES:
                    SOURCE_VARIABLE_NAME: DESTINATION_VARIABLE_NAME
```

3.2 Run Simulation

3.2.1 Local python

To run in locally you can just run the python file you created.

3.2.2 Docker

To run with Docker you first need to create an image with:

```
docker build . -t nestli
```

This is further described in the installation section.

You run a container specifying your source file path with PATH_TO_PROJECT_FOLDER with:

```
docker run -it -v "PATH_TO_PROJECT_FOLDER:/example_folder" nestli
```

This folder must contain a python file called nestli_example_run.py and your config file and other files your simulation needs. It will be copied to the container and the results will get copied back to it.

CHAPTER

FOUR

INDICES AND TABLES

- modindex
- search