agentMET4FOF Documentation

Bang Xiang Yong

Jul 29, 2021

Getting started:

CHAPTER 1

Multi-Agent System for IIoT

1.1 Key facts

- *FOSS project*
- allows to
  - quickly set up and run a *metrologically enabled multi-agent system*
  - *handle both static data sets and online data streams*
  - *consider measurement uncertainties as well as metadata with the provided message system*
- *installable as a Python package or ready-to-deploy Docker image*
- comes bundled with *several introductary and advanced tutorials*
- accompanied by several use cases with close-to-industry IIoT applications in our GitHub organisation
- comprehensive and ever-growing *documentation*

1.2 Table of content

1.3 Quickstart

agentMET4FOF comes bundled with several *tutorials* to get you started as quick as possible. In your Python console execute the following to run the first tutorial.

```
>>> from agentMET4FOF_tutorials.tutorial_1_generator_agent import demonstrate_
↪generator_agent_use
>>> generator_agent_network = demonstrate_generator_agent_use()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

|-----------------------------------------------------------|
|                                                           |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://0.0.0.0:8050/ |
|                                                           |
|-----------------------------------------------------------|

INFO [2021-02-05 18:12:52.277759] (SineGeneratorAgent_1): INITIALIZED
INFO [2021-02-05 18:12:52.302862] (MonitorAgent_1): INITIALIZED
[2021-02-05 18:12:52.324078] (SineGeneratorAgent_1): Connected output module:␣
↪MonitorAgent_1
SET STATE:   Running
[...]
```

```
>>> generator_agent_network.shutdown()
0
NS shut down.
```

1.4 About

1.4.1 Features

Some notable features of agentMET4FOF include :

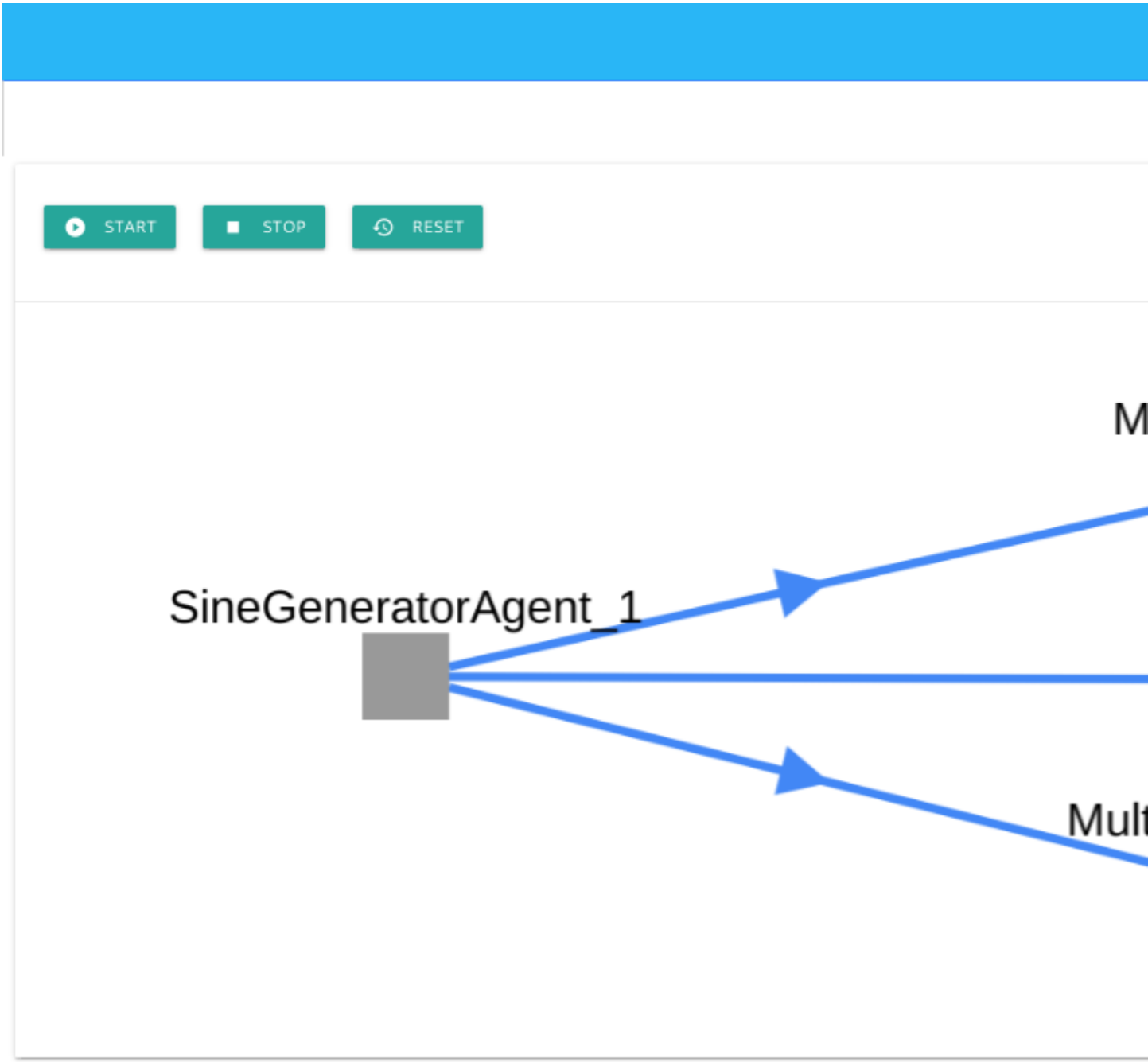
- Modular agent classes for metrological data streams and analytics
- A built-in buffering mechanism to decouple transmission, processing and visualization of data
- Easy connection among software agents to send and receive data
- Choose backends between:
  - *Osbrain* for simulating as well as handling real distributed systems running Python connected via a TCP network, and

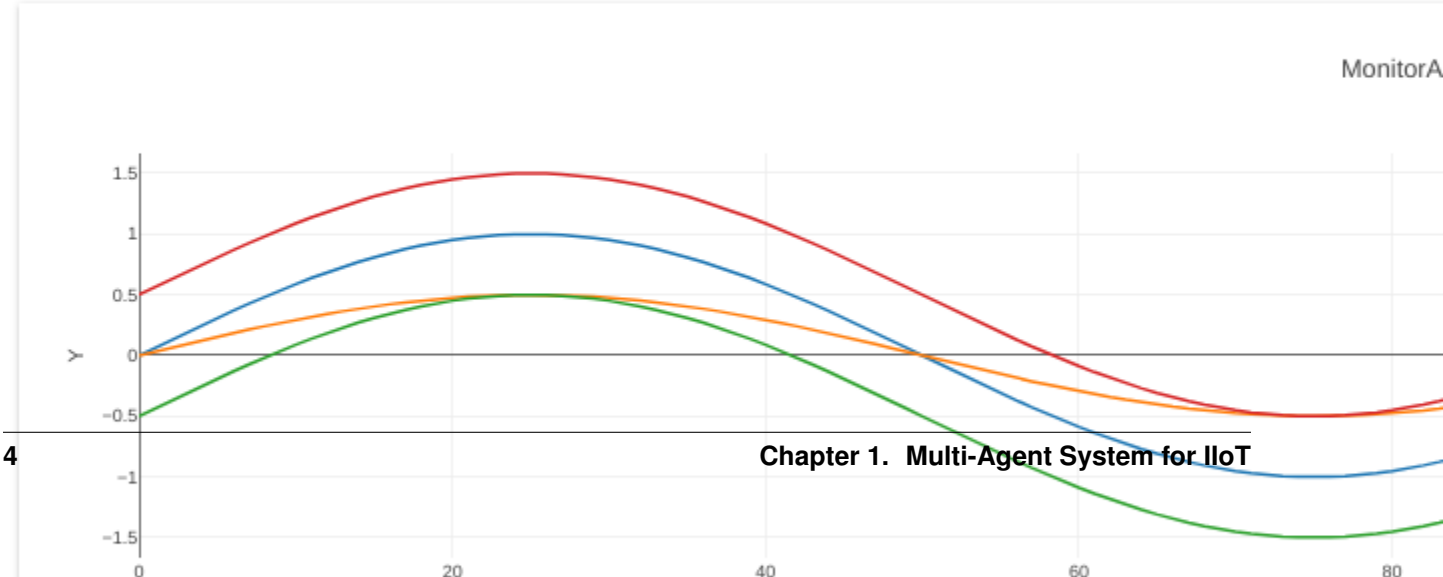
  - *Mesa* for local simulations of distributed systems, debugging and more high-performance execution
- Interactive and customisable dashboard from the get-go to:
  - Visualize and change agent-network topologies
  - Visualize groups of cooperative agents as *Coalitions*
  - View and change the agents' parameters
  - View the agents' outputs as plotly or matplotlib plots or generate and embed your own images

1.5 The agentMET4FOF dashboard

agentMET4FOF comes bundled with our so called *dashboard*. It is an optional component of every agent network and provides a web browser based view. You can observe the state of your agents, modify the connections between them and even add more pre-made agents to your network all during run-time. The address to your dashboard is printed to the console on every launch of an agent network.

The following image is close to what you will find in your browser on execution of tutorial 2. For details on the tutorials visit our *video tutorial series*.


START
STOP
RESET
SineGeneratorAgent_1
M
Mult


MonitorA
1.5
1
0.5
0
−0.5
−1
−1.5
Y
0
20
40
60
80

1.6 Tutorials

As mentioned above, agentMET4FOF comes bundled with several tutorials to get you started as quick as possible. You will find tutorials on how to set up:

- a simple pipeline to plot a signal
- a simple pipeline with signal postprocessing
- an advanced pipeline with multichannel signals
- a simple metrological datastream
- pipelines to determine redundancy in sensor networks

. . . and more!

1.7 Documentation and screencasts

Extended documentation can be found on ReadTheDocs.

1.7.1 Screencast series

Additionally, we provide some screencasts based on agentMET4FOF 0.4.1 on the project homepage in the section *Tutorials for the multi-agent system agentMET4FOF*. You can self-register on the linked page and get started immediately. The video series begins with our motivation for creating agentMET4FOF, guide you through the installation of Python and other recommended software until you execute the tutorials on your machine.

1.7.2 Live online tutorial during early development

In an early development stage we held a live online tutorial based on agentMET4FOF 0.1.0 which you can download.

If questions arise, or you feel something is missing, reach out to us.

1.8 Installation

There are different ways to run agentMET4FOF. Either:

1. you install Python and our package agentMET4FOF in a virtual Python environment on your computer, or
2. you install Docker, start agentMET4FOF in a container and visit the Jupyter Notebook server and the agentMET4FOF dashboard directly in your browser or even deploy it over a proper webserver.

In the *video tutorials series* we guide you through every step of option 1. More detailed instructions on both options you can find in the installation section of the docs.

1.9 Contributing

Whenever you are involved with agentMET4FOF, please respect our Code of Conduct . If you want to contribute back to the project, after reading our Code of Conduct, take a look at our open developments in the project board , pull requests and search the issues . If you find something similar to your ideas or troubles, let us know by leaving

a comment or remark. If you have something new to tell us, feel free to open a feature request or bug report in the issues. If you want to contribute code or improve our documentation, please check our contributing guide.

1.10 Coming soon

- Improved handling of metadata
- More advanced signal processing

For a comprehensive overview of current development activities and upcoming tasks, take a look at the project board, issues and pull requests.

1.11 Citation

If you publish results obtained with the help of agentMET4FOF, please cite the linked .

1.12 Acknowledgement

This work was part of the Joint Research Project Metrology for the Factory of the Future (Met4FoF), project number 17IND12 of the European Metrology Programme for Innovation and Research (EMPIR). The EMPIR is jointly funded by the EMPIR participating countries within EURAMET and the European Union.

1.13 Disclaimer

This software is developed as a joint effort of several project partners namely:

- Institute for Manufacturing of the University of Cambridge (IfM)
- Physikalisch-Technische Bundesanstalt (PTB)
- Van Swinden Laboratory (VSL)
- National Physics Laboratory (NPL)

under the lead of IfM. The software is made available “as is” free of cost. The authors and their institutions assume no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, safety, suitability or any other characteristic. In no event will the authors be liable for any direct, indirect or consequential damage arising in connection with the use of this software.

1.14 ©License

CHAPTER 2

Installation of agentMET4FOF

As already mentioned in the README, agentMET4FOF can either be *installed as a Python package* or *launched in a Docker container*.

2.1 Install agentMET4FOF as a Python package

The installation of agentMET4FOF is as straightforward as the Python ecosystem suggests. In the video tutorial series linked in the README we guide you through every step until you have agentMET4FOF running on your machine.

If you want to take the steps manually we guide you through in this document.

2.1.1 Set up a virtual environment

For the motivation of creating a virtual environment for your installation of the agents check the official Python docs on that topic . You have the option to do this with *Anaconda*, if you already have it installed, or use the Python built-in tool `venv`. The commands differ slightly between *Windows* and *Mac/Linux* or if you use *Anaconda* .

Create a `venv` Python environment on Windows

In your Windows PowerShell execute the following to set up a virtual environment in a folder of your choice.

```
PS C:> cd C:\LOCAL\PATH\TO\ENVS
PS C:\LOCAL\PATH\TO\ENVS> py -3 -m venv agentMET4FOF_venv
PS C:\LOCAL\PATH\TO\ENVS> agentMET4FOF_venv\Scripts\activate
```

Proceed to *the next step*.

Create a `venv` Python environment on Mac & Linux

In your terminal execute the following to set up a virtual environment in a folder of your choice.

```
$ cd /LOCAL/PATH/TO/ENVS
$ python3 -m venv agentMET4FOF_venv
$ source agentMET4FOF_venv/bin/activate
```

Proceed to *the next step*.

Create an Anaconda Python environment

To get started with your present *Anaconda* installation just go to *Anaconda prompt* and execute

```
$ cd /LOCAL/PATH/TO/ENVS
$ conda env create --file /LOCAL/PATH/TO/agentMET4FOF/environment.yml
```

That's it!

2.1.2 Install agentMET4FOF via `pip`

Once you activated your virtual environment, you can install agentMET4FOF via:

```
pip install agentMET4FOF
```

```
Collecting agentMET4FOF
[...]
Successfully installed agentMET4FOF-[...] [...]
```

That's it!

2.2 Use agentMET4FOF from inside a Docker container

Every version of agentMET4FOF since v0.9.1dev is additionally accompanied by a so-called Docker image. With its help, agentMET4FOF can be launched quickly on any computer with a Docker installation without installing Python. agentMET4FOF can then be used directly in the browser using the supplied or your own Jupyter notebooks and even the dashboard can be visited in the browser after its launch. The following steps are required for this.

1. *Install Docker*
2. *Download and import the agentMET4FOF Docker image*
3. a) *Start a container from the image for local use*

   b) *Deploy the containerized agents via a webserver*

2.2.1 Install Docker

The official Docker documentation guides you through. Please continue with *the next step*, once you completed the Docker installation.

2.2.2 Download and import the agentMET4FOF Docker image

You can download the *Docker image for agentMET4FOF Jupyter Notebook server* as one of the release assets, import it locally with

```
> docker load -i LOCAL\PATH\TO\DOWNLOADS\tagged_docker_image_agentMET4FOF_jupyter.tar.
↪gz
```

2.2.3 Start a container from the image for local use

After importing the image, you can launch it straight away with

```
> docker run -p 8888:8888 -p 8050:8050 --rm agentmet4fof
```

In this command's output you will find the usual Jupyter Notebook token URL, which you can open in your browser. After starting an agent network in one of the tutorials or your own notebooks, you will find the dashboard URL in the notebook's output resembling something like `http://0.0.0.0:8050`.

2.2.4 Deploy the containerized agents via a webserver

To make the agents accessible via a TCP/IP network such as the internet using a web server like Nginx, all that is required is the correct webserver configuration. The example configuration presented here will spawn the container so that the Jupyter Notebook server is accessible via http://agent.domain.com, and the dashboard is accessible after start-up under http://agent.domain.com/YOUR_FOLDER_NAME_OF_CHOICE.

Start the container with the dashboard at a subfolder

Launch the container with

```
> docker run -p 8888:8888 -p 8050:8050 --rm  \
--env DASH_URL_BASE_PATHNAME=/YOUR_FOLDER_NAME_OF_CHOICE/ agentmet4fof
```

This ensures, that the dashboard will be reachable under whatever domain you are using followed by `/YOUR_FOLDER_NAME_OF_CHOICE`.

Configure the Nginx

We assume you are confident in using Nginx in general. The corresponding configuration should contain the following server blocks to ensure the Jupyter Notebook server is working, as well as the dashboard

```
server {
        server_name agent.domain.com;

        location / {
            proxy_pass http://127.0.0.1:8888;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

            # websocket headers
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection $connection_upgrade;
            proxy_set_header X-Scheme $scheme;

            proxy_buffering off;
```

(continues on next page)

(continued from previous page)

```
        }

        location /YOUR_FOLDER_NAME_OF_CHOICE/ {
            proxy_pass http://127.0.0.1:8050/YOUR_FOLDER_NAME_OF_CHOICE/;
        }

    [...]

}
```

CHAPTER 3

Changelog

3.1 v0.11.0 (2021-07-29)

3.1.1 Feature

- **Plots:** Adapt dashboard such that all agents of types `MonitorAgent` and `MetrologicalMonitorAgent` get plotted regardless of their name (`67a2544`)
- **network:** Introduce a method `agents_by_type` to query for agents of a given type (`e43a5b9`)

3.1.2 Fix

- Allow spaces in agents' names which get replaced by underscores for the *osBrain* backend (`2d8437d`)

3.1.3 Refactor

- **AgentNetwork:** Change some of the internal variable (`06c9a36`)
- **SineGeneratorAgent:** Improve data sending by including actually computed time (`87c3e34`)
- **signal_agents:** Introduce "official" _sine_stream instance variable for SineGeneratorAgent (`98d0f19`)
- **signal_agents:** Reintroduce NoiseAgent placeholder after module refactorization (`f97f4d9`)

3.1.4 Documentation

- **network:** Introduce some more type hints (`ee4b158`)

See all commits in this version

3.2 v0.10.1 (2021-07-23)

3.2.1 Fix

- **CODE_OF_CONDUCT:** Finally add a code of conduct (56ee503)

3.2.2 Refactor

- **CONTRIBUTING:** Change heading and location of *CONTRIBUTING.md* to make it more visible (1ff173f)

3.2.3 Documentation

- **README:** Introduce key facts (0e2d936)
- **README:** Introduce Tutorials section (e1f41e9)
- **README:** Introduce Contributing section (85e8d9f)

See all commits in this version

3.3 v0.10.0 (2021-07-21)

3.3.1 Feature

- **Docker:** Create container image for every release (1a40eaf)

3.3.2 Refactor

- **network:** Change delivery ip to 0.0.0.0, i.e. all interfaces instead of 127.0.0.1 to enable docker deployment (2472826)

3.3.3 Documentation

- **ReadTheDocs:** Include Docker guide (7c35bba)
- **CONTRIBUTING:** Mention commit type refactor in contributing guide (a4891e7)
- **metrological_redundancy_agents:** Introduce redundancy agent into docs (cb493cb)

See all commits in this version

3.4 v0.9.0 (2021-07-08)

3.4.1 Feature

- **agents:** Adds `ampl` and `phase_ini` parameters to SineGeneratorAgent (774a91b)
- **streams:** Adds `ampl` and `phase_ini` parameters to SineGenerator and CosineGenerator (998b158)
- **metrological_streams:** Adds `ampl` and `phase_ini` parameters to MetrologicalSineGenerator (fc9901e)

- **redundancy agent:** Adds redundancy agent class to metrological_agents.py (7fbe934)
- **MetrologicalGeneratorAgent:** Add an agent to handle generic metrological data streams (3334762)
- **MET4FoF redundancy:** Method init_parameters1 removed. removed unnecessary comments (afe47c3)
- **MET4FoF redundancy:** Second redundancy tutorial integrated (c527f33)
- **MET4FoF redundancy:** Tutorial 7 added (caa62ca)
- **MET4FoF redundancy:** Met4fof redundancy modules added (bb3cb6a)
- **multiwave:** Added multiwave generator to metrological_streams.py (7b0bf4e)
- **multiwave:** Added multiwave generator to metrological_streams.py (035d7c9)

3.4.2 Fix

- **tutorial 3:** Replace one of the generator agents' streams by the cosine stream (2f20183)
- **metrological_agents:** AgentBuffer import was accidentally removed (cbec978)

3.4.3 Refactor

- **agents and metrological_streams and streams:** Renames 'ampl' to 'amplitude' and 'phase_ini' to 'initial_phase' (d0d8271)
- **metrological_agents:** Refactor metrological_agents.py into a package metrological_agents and include classes into agents (5558008)
- **metrological_streams:** Refactor metrological_streams.py into a package metrological_streams and include classes into streams (9da3744)
- **base_streams:** Refactor base_streams.py by applying black (34ff2d0)
- **signal_streams:** Introduce **all** variable into signal_streams.py (8c62972)
- **signal_streams:** Refactor signal_streams.py by applying black (e604418)
- **streams:** Refactor streams.py into a package streams with the modules base_streams and signal_streams (11368ee)
- **signal_agents:** Introduce **all** variable into signal_agents.py (364702f)
- **signal_agents:** Refactor signal_agents.py by applying black (396a95e)
- **base_agents:** Introduce **all** variable into base_agents.py (34f626b)
- **base_agents:** Refactor base_agents.py by applying black and shortening line lengths (8173324)
- **agents:** Refactor agents.py into a package agents with the modules base_agents and signal_agents (c7bdfae)
- **multi_generator:** Replaces `amplitude` with `value` in variable names in generator classes (ee45e1c)
- **simple_generator:** Replaces `amplitude` with `value` in variable names in generator classes (f7475ba)
- **metrological agents and streams:** Make imports relative (907cdb2)
- **streams:** Replaces `amplitude` with `value` in variable names in generator classes (177b231)
- **metrological_streams:** Replaces `amplitude` with `value` in variable names (b4a0118)
- **metrological_streams:** Reorder parameters (b0c465f)
- **MetrologicalDataStreamMet4FoF:** Refine MetrologicalDataStreamMet4FoF docstring (1bfc2a6)

3.4.4 Documentation

- **ReadTheDocs:** Adapt documentation to new package and module structure (`ab207c4`)
- **tutorial 3:** Update fixed notebook with most recent output (`5511079`)
- **CONTRIBUTING:** Change commit log examples to `develop`'s commits (`9ee020f`)
- **CHANGELOG:** Insert actually current changelog into docs (`aff2f6a`)

See all commits in this version

3.5 v0.8.1 (2021-06-10)

See all commits in this version

3.6 v0.8.0 (2021-06-10)

3.6.1 Feature

- **MetrologicalGeneratorAgent:** Rewrote to default metrological generator agent for both streams (`52606d7`)
- **SineGeneratorAgent:** Added default sine generator agent to metrological_agents.py (`54f4035`)
- **multiwave_generator:** Multiwave generator added to metrological_streams.py (`682a5ff`)

3.6.2 Documentation

- **tutorial_7:** Introduce tutorial 7 into ReadTheDocs (`01d7e44`)
- **tutorial_default_generator:** Add a tutorial for a generic metrologically enabled agent (`7bebef1`)

See all commits in this version

3.7 v0.7.0 (2021-06-04)

3.7.1 Feature

- **enhance_ui:** Button to export agent network display as png (`3b41610`)
- **enhance_subscribe:** Allow special channels for requests (`a88a817`)

3.7.2 Fix

- **enhance_UI:** Fix bug on callback of `agents-network.generateImage` (`d9125ab`)
- **enhance_UI:** Fix importing name for UI example (`58b990b`)
- **enhance_ui:** Required import visdcc (`f02429e`)
- **enhance_subscribe:** Specify the plot channel for tests (`1d293e7`)

3.7.3 Documentation

- **enhance_UI:** Refactor agent_type to agent_name_filter (a6f8da9)
- **enhance_UI:** Add docstring for create_edges_cytoscape method (b66fcb3)
- **enhance_UI:** Intuitive name for toast message function in UI (c155138)
- **enhance_UI:** Rename and add description to UI example (3a03428)
- **enhance_UI:** Refactor tutorial folder on UI example (360f6da)

See all commits in this version

3.8 v0.6.4 (2021-05-06)

3.8.1 Fix

- **streams:** Remove unnecessary instantiation of sine_freq in method sine_wave_function (192a6c9)

See all commits in this version

3.9 v0.6.3 (2021-04-28)

3.9.1 Fix

- **time_series_buffer:** Resolve storing nan values (bbc1ae9)

3.9.2 Documentation

- **metrological_agents:** Reformat two of our docstrings to properly display thos example values in the docs (c56cca9)
- **metrological_agents:** Correct docstrings for `_concatenate()` (eb6a20b)

See all commits in this version

3.10 v0.6.2 (2021-03-19)

3.10.1 Fix

- **dashboard:** Finally resolve #186 by cleanly shutting down dashboard (9903d15)

3.10.2 Documentation

- **CHANGELOG:** Include CHANGELOG into docs (4ba20d3)

See all commits in this version

3.11 v0.6.1 (2021-03-12)

3.11.1 Fix

- **metrological_streams:** Fixed computational error for batches of size more than one (`686bad5`)
- **agents:** Fixed `buffer.clear` method, which did not work anymore after moving from `memory` to `buffer`

3.11.2 Documentation

- **README:** Include the Zenodo DOI banner and add the Citation section (`4a57dd8`)
- **docstrings:** Seriously improve docstrings and type hinting in `agents.py` and `dashboard/LayoutHelper.py`

See all commits in this version

CHAPTER 4

Contributor Covenant Code of Conduct

4.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

4.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

4.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at https://github.com/Met4FoF/agentMET4FOF/graphs/contributors. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

4.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

4.6.1 1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

4.6.2 2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

4.6.3 3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4.6.4 4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

4.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

CHAPTER 5

Contribute to agentMET4FOF

Whenever you are involved with agentMET4FOF, please respect our Code of Conduct . If you want to contribute back to the project, after reading our Code of Conduct, take a look at our open developments in the project board , pull requests and search the issues . If you find something similar to your ideas or troubles, let us know by leaving a comment or remark. If you have something new to tell us, feel free to open a feature request or bug report in the issues.

If you want to contribute code or improve our documentation, we provide this guide to get the desired system configuration aligned with our development environments. The code you produce should be seamlessly integrable into agentMET4FOF by aligning your work with the established workflows. This guide should work on all platforms and provide everything needed to start developing for agentMET4FOF. Please open an issue or ideally contribute to this guide as a start, if problems or questions arise.

5.1 Guiding principles

The agentMET4FOF development process is based on the following guiding principles:

- actively maintain, ensuring security vulnerabilities or other issues are resolved in a timely manner
- employ state-of-the-art development practices and tools, specifically
  - follow semantic versioning
  - use conventional commit messages
  - consider the PEP8 style guide, wherever feasible

5.1.1 Get the code on GitHub and locally

For collaboration, we recommend forking the repository as described here. Simply apply the changes to your fork and open a Pull Request on GitHub as described here. For small changes it will be sufficient to just apply your changes on GitHub and send the PR right away. For more comprehensive work, you should clone your fork and read on carefully.

5.1.2 Initial development setup

This guide assumes you already have a valid runtime environment for agentMET4FOF as described in the Installation guide.

First install the known to work configuration of our dependencies into you virtual environment:

```
(agentMET4FOF_venv) $ pip install -r requirements.txt -r dev-requirements.txt
```

5.1.3 Advised toolset

If you followed the steps for the *initial development setup* you have everything at your hands:

- *Sphinx* for automated generation of our documentation on ReadTheDocs
- *pytest* as testing framework backed by *hypothesis* and *coverage*.
- *python-semantic-release* in
- our pipeline on *CircleCI* . All requirements for contributions are derived from this.

5.1.4 Coding style

As long as the readability of mathematical formulations is not impaired, our code shoulds follow PEP8. We know we can improve on this requirement for the existing code base as well, but all code added should already conform to PEP8. For automating this uniform formatting task we use the Python package *black*. It is easy to handle and integrable into most common IDEs, such that it is automatically applied.

5.1.5 Commit messages

agentMET4FOF commit messages follow some conventions to be easily human and machine-readable.

Commit message structure

Conventional commit messages are required for the following:

- Releasing automatically according to semantic versioning
- Generating a changelog automatically

Parts of the commit messages and links appear in the changelogs of subsequent releases as a result. We use the following types:

- *feat*: for commits that introduce new features (this correlates with MINOR in semantic versioning)
- *docs*: for commits that contribute significantly to documentation
- *fix*: commits in which bugs are fixed (this correlates with PATCH in semantic versioning)
- *test*: Commits that apply significant changes to tests
- *chore*: Commits that affect other non-PyDynamic components (e.g. ReadTheDocs, Git , . . .)
- *revert*: commits, which undo previous commits using `git revert`
- *refactor*: commits, which contain to refactoring activities

- *wip*: Commits which are not recognizable as one of the above-mentioned types until later, usually during a PR merge. The merge commit is then marked as the corresponding type.

Of the types mentioned above, the following appear in separate sections of the changelog:

- *Feature*: *feat*
- *Documentation*: *docs*
- *Fix*: *fix*
- *Test*: *test*

Commit message styling

Based on established community standards, the first line of a commit message should complete the following sentence:

> If this commit is applied, it will...

More comprehensive messages should contain an empty line after that and everything else needed starting from the third line. Each line should not exceed 100 characters.

BREAKING CHANGEs

Since agentMET4FOF is not yet considered stable, we do not mark BREAKING CHANGES. As a consequence, at any time commits may changes parts of agentMET4FOF's public interface so that previously written code may no longer be executable. If this occurs we try though, to mention migration strategies in the corresponding release descriptions.

Commit message examples

For examples please checkout the Git Log.

5.1.6 Testing

We strive to increase our code coverage with every change introduced. This requires that every new feature and every change to existing features is accompanied by appropriate *pytest* testing. We test the basic components for correctness and, if necessary, the integration into the big picture. It is usually sufficient to create appropriately named methods in one of the existing modules in the subfolder test. If necessary add a new module that is appropriately named.

5.2 Workflow for adding completely new functionality

In case you add a new feature you generally follow the pattern:

- read through and follow this contribution advices and tips, especially regarding the *advised tool* set and *coding style*
- open an according issue to submit a feature request and get in touch with other agentMET4FOF developers and users
- fork the repository or update the *develop* branch of your fork and create an arbitrary named feature branch from *develop*
- decide which package and module your feature should be integrated into

- if there is no suitable package or module, create a new one and a corresponding module in the *tests* subdirectory with the same name prefixed by *test_*
- if new dependencies are introduced, add them to *setup.py* or *dev-requirements.in*
- during development write tests in alignment with existing test modules, for example *test_addremove_metrological_agents*
- write docstrings in the NumPy docstring format
- as early as possible create a draft pull request onto the upstream's *develop* branch
- once you think your changes are ready to merge, request a review from the *agentMET4FOF collaborators* (you will find them in the according drop-down) and mark your PR as *ready for review*
- at the latest now you will have the opportunity to review the documentation automatically generated from the docstrings on ReadTheDocs after your reviewers will set up everything
- resolve the conversations and have your pull request merged

5.3 Documentation

The documentation of agentMET4FOF consists of three parts. Every adaptation of an existing feature and every new feature requires adjustments on all three levels:

- user documentation on ReadTheDocs
- examples in the form of Jupyter notebooks for extensive features and Python scripts for features which can be comprehensively described with few lines of commented code
- developer documentation in the form of comments in the code

5.3.1 User documentation

To locally generate a preview of what ReadTheDocs will generate from your docstrings, you can simply execute after activating your virtual environment:

```
(agentMET4FOF_venv) $ sphinx-build docs/ docs/_build
Sphinx v3.1.1 in Verwendung
making output directory...
[...]
build abgeschlossen.

The HTML pages are in docs/_build.
```

After that you can open the file _./docs/*build/index.html* relative to the project's root with your favourite browser. Simply re-execute the above command after each change to the docstrings to update your local version of the documentation.

5.3.2 Examples

We want to provide extensive sample material for all agentMET4FOF features in order to simplify the use or even make it possible in the first place. We collect the examples in the subfolder agentMET4FOF_tutorials.

5.3.3 Comments in the code

Regarding comments in the code we recommend to invest 45 minutes for the PyCon DE 2019 Talk of Stefan Schwarzer, a 20+-years Python developer: Commenting code - beyond common wisdom.

5.4 Manage dependencies

We use *pip-tools* for dependency management. The root folder contains a *requirements.txt* and a *dev-requirements.txt* for the supported Python version. *pip-tools*' command `pip-compile` finds the right versions from the dependencies listed in *setup.py* and the *dev-requirements.in* and is manually run by the maintainers regularly.

5.5 Licensing

All contributions are released under agentMET4FOF's GNU Lesser General Public License v3.0.

CHAPTER 6

Learning how to use the agents

6.1 Tutorial 1 - A simple pipeline to plot a signal

First we define a simple pipeline of two agents, of which one will generate a signal (in our case a *SineGeneratorAgent*) and the other one plots the signal on the dashboard (this is always a *MonitorAgent*).

We define a *SineGeneratorAgent* for which we have to override the functions `init_parameters()` & `agent_loop()` to define the new agent's behaviour.

- `init_parameters()` is used to setup the input data stream and potentially other necessary parameters.
- `agent_loop()` will be endlessly repeated until further notice. It will sample by sample extract the input data stream's content and push it to all agents connected to *SineGeneratorAgent*'s output channel by invoking `send_output()`.

The *MonitorAgent* is connected to the *SineGeneratorAgent*'s output channel and per default automatically plots the output.

Each agent has an internal `current_state` which can be used as a switch to change the behaviour of the agent. The available states are listed here.

As soon as all agents are initialized and the connections are set up, the agent network is started by accordingly changing all agents' state simultaneously.

[1]:
```
# %load tutorial_1_generator_agent.py
from agentMET4FOF.agents import AgentMET4FOF, AgentNetwork, MonitorAgent
from agentMET4FOF.streams import SineGenerator


class SineGeneratorAgent(AgentMET4FOF):
    """An agent streaming a sine signal

    Takes samples from the :py:mod:`SineGenerator` and pushes them sample by sample
    to connected agents via its output channel.
    """
```

(continues on next page)

(continued from previous page)

```
    # The datatype of the stream will be SineGenerator.
    _sine_stream: SineGenerator

    def init_parameters(self):
        """Initialize the input data

        Initialize the input data stream as an instance of the
        :py:mod:`SineGenerator` class
        """
        self._sine_stream = SineGenerator()

    def agent_loop(self):
        """Model the agent's behaviour

        On state *Running* the agent will extract sample by sample the input data
        streams content and push it via invoking :py:method:`AgentMET4FOF.send_output`.
        """
        if self.current_state == "Running":
            sine_data = self._sine_stream.next_sample()  # dictionary
            self.send_output(sine_data["quantities"])


def demonstrate_generator_agent_use():
    # Start agent network server.
    agent_network = AgentNetwork()

    # Initialize agents by adding them to the agent network.
    gen_agent = agent_network.add_agent(agentType=SineGeneratorAgent)
    monitor_agent = agent_network.add_agent(agentType=MonitorAgent)

    # Interconnect agents by either way:
    # 1) by agent network.bind_agents(source, target).
    agent_network.bind_agents(gen_agent, monitor_agent)

    # 2) by the agent.bind_output().
    gen_agent.bind_output(monitor_agent)

    # Set all agents' states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution
    return agent_network


if __name__ == "__main__":
    demonstrate_generator_agent_use()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
```

(continues on next page)

(continued from previous page)

```
|                                                                |
----------------------------------------------------------------

INFO [2021-02-05 19:13:45.925758] (SineGeneratorAgent_1): INITIALIZED
INFO [2021-02-05 19:13:45.960173] (MonitorAgent_1): INITIALIZED
[2021-02-05 19:13:45.975418] (SineGeneratorAgent_1): Connected output module:␣
↪MonitorAgent_1
SET STATE:   Running
[2021-02-05 19:13:46.932922] (SineGeneratorAgent_1): Pack time: 0.000957
[2021-02-05 19:13:46.935938] (SineGeneratorAgent_1): Sending: [0.]
[2021-02-05 19:13:46.939035] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.]), 'senderType': 'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:13:46.941337] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.])}
[2021-02-05 19:13:46.942394] (MonitorAgent_1): Tproc: 0.002278
[2021-02-05 19:13:47.932181] (SineGeneratorAgent_1): Pack time: 0.000614
[2021-02-05 19:13:47.935312] (SineGeneratorAgent_1): Sending: [0.06279052]
[2021-02-05 19:13:47.936553] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.06279052]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:47.941367] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052])}
[2021-02-05 19:13:47.942190] (MonitorAgent_1): Tproc: 0.004357
[2021-02-05 19:13:48.931650] (SineGeneratorAgent_1): Pack time: 0.00047
[2021-02-05 19:13:48.933397] (SineGeneratorAgent_1): Sending: [0.12533323]
[2021-02-05 19:13:48.934297] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.12533323]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:48.936482] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323])}
[2021-02-05 19:13:48.936997] (MonitorAgent_1): Tproc: 0.002201
[2021-02-05 19:13:49.932143] (SineGeneratorAgent_1): Pack time: 0.000937
[2021-02-05 19:13:49.940442] (SineGeneratorAgent_1): Sending: [0.18738131]
[2021-02-05 19:13:49.934471] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.18738131]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:49.938767] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131])}
[2021-02-05 19:13:49.939977] (MonitorAgent_1): Tproc: 0.004969
[2021-02-05 19:13:50.930904] (SineGeneratorAgent_1): Pack time: 0.000255
[2021-02-05 19:13:50.931636] (SineGeneratorAgent_1): Sending: [0.24868989]
[2021-02-05 19:13:50.932383] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.24868989]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:50.933944] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989])}
[2021-02-05 19:13:50.934185] (MonitorAgent_1): Tproc: 0.001522
NS shut down.
[2021-02-05 19:13:51.932632] (SineGeneratorAgent_1): Pack time: 0.001127
[2021-02-05 19:13:51.935690] (SineGeneratorAgent_1): Sending: [0.30901699]
[2021-02-05 19:13:51.935544] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.30901699]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:51.944128] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699])}
[2021-02-05 19:13:51.944378] (MonitorAgent_1): Tproc: 0.008396
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:13:52.930087] (SineGeneratorAgent_1): Pack time: 0.000108
[2021-02-05 19:13:52.930343] (SineGeneratorAgent_1): Sending: [0.36812455]
[2021-02-05 19:13:52.930548] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.36812455]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:52.930905] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455])}
[2021-02-05 19:13:52.931004] (MonitorAgent_1): Tproc: 0.000381
[2021-02-05 19:13:53.930135] (SineGeneratorAgent_1): Pack time: 0.000144
[2021-02-05 19:13:53.930526] (SineGeneratorAgent_1): Sending: [0.42577929]
[2021-02-05 19:13:53.930537] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.42577929]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:53.930940] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929])}
[2021-02-05 19:13:53.930993] (MonitorAgent_1): Tproc: 0.00038
[2021-02-05 19:13:54.932072] (SineGeneratorAgent_1): Pack time: 0.0009
[2021-02-05 19:13:54.934391] (SineGeneratorAgent_1): Sending: [0.48175367]
[2021-02-05 19:13:54.934275] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.48175367]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:54.936874] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929, 0.48175367])}
[2021-02-05 19:13:54.937315] (MonitorAgent_1): Tproc: 0.002635
[2021-02-05 19:13:55.931558] (SineGeneratorAgent_1): Pack time: 0.000454
[2021-02-05 19:13:55.933193] (SineGeneratorAgent_1): Sending: [0.53582679]
[2021-02-05 19:13:55.933792] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.53582679]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:13:55.936039] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929, 0.48175367, 0.53582679])}
[2021-02-05 19:13:55.936606] (MonitorAgent_1): Tproc: 0.002345
```

6.2 Tutorial 2 - A simple pipeline with signal postprocessing.

Here we demonstrate how to build a *MathAgent* as an intermediate to process the *SineGeneratorAgent*'s output before plotting. Subsequently, a *MultiMathAgent* is built to show the ability to send a dictionary of multiple fields to the recipient. We provide a custom `on_received_message()` function, which is called every time a message is received from input agents.

The received message is a dictionary of the form:

```
{
'from':agent_name,
'data': data,
'senderType': agent_class_name,
'channel':'channel_name'
}
```

By default, `'channel'` is set to `"default"`, however a custom channel can be set when needed, which is demonstrated in the next tutorial.

[1]:
```
# %load tutorial_2_math_agent.py
from agentMET4FOF.agents import (
    AgentMET4FOF,
    AgentNetwork,
    MonitorAgent,
    SineGeneratorAgent,
)


class MathAgent(AgentMET4FOF):
    def on_received_message(self, message):
        data = self.divide_by_two(message["data"])
        self.send_output(data)

    # Define simple math functions.
    @staticmethod
    def divide_by_two(numerator: float) -> float:
        return numerator / 2


class MultiMathAgent(AgentMET4FOF):

    _minus_param: float
    _plus_param: float

    def init_parameters(self, minus_param=0.5, plus_param=0.5):
        self._minus_param = minus_param
        self._plus_param = plus_param

    def on_received_message(self, message):
        minus_data = self.minus(message["data"], self._minus_param)
        plus_data = self.plus(message["data"], self._plus_param)

        self.send_output({"minus": minus_data, "plus": plus_data})

    @staticmethod
    def minus(minuend: float, subtrahend: float) -> float:
        return minuend - subtrahend

    @staticmethod
    def plus(summand_1: float, summand_2: float) -> float:
        return summand_1 + summand_2


def main():
    # start agent network server
    agentNetwork = AgentNetwork()
    # init agents
    gen_agent = agentNetwork.add_agent(agentType=SineGeneratorAgent)
    math_agent = agentNetwork.add_agent(agentType=MathAgent)
    multi_math_agent = agentNetwork.add_agent(agentType=MultiMathAgent)
    monitor_agent = agentNetwork.add_agent(agentType=MonitorAgent)
    # connect agents : We can connect multiple agents to any particular agent
    agentNetwork.bind_agents(gen_agent, math_agent)
    agentNetwork.bind_agents(gen_agent, multi_math_agent)
    # connect
    agentNetwork.bind_agents(gen_agent, monitor_agent)
```

(continues on next page)

(continued from previous page)

```
    agentNetwork.bind_agents(math_agent, monitor_agent)
    agentNetwork.bind_agents(multi_math_agent, monitor_agent)
    # set all agents states to "Running"
    agentNetwork.set_running_state()

    # allow for shutting down the network after execution
    return agentNetwork


if __name__ == "__main__":
    main()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-02-05 19:18:07.585019] (SineGeneratorAgent_1): INITIALIZED
INFO [2021-02-05 19:18:07.619684] (MathAgent_1): INITIALIZED
INFO [2021-02-05 19:18:07.654192] (MultiMathAgent_1): INITIALIZED
INFO [2021-02-05 19:18:07.691566] (MonitorAgent_1): INITIALIZED
[2021-02-05 19:18:07.706815] (SineGeneratorAgent_1): Connected output module:␣
↪MathAgent_1
[2021-02-05 19:18:07.714664] (SineGeneratorAgent_1): Connected output module:␣
↪MultiMathAgent_1
[2021-02-05 19:18:07.724529] (SineGeneratorAgent_1): Connected output module:␣
↪MonitorAgent_1
[2021-02-05 19:18:07.738462] (MathAgent_1): Connected output module: MonitorAgent_1
[2021-02-05 19:18:07.750583] (MultiMathAgent_1): Connected output module:␣
↪MonitorAgent_1
SET STATE:   Running
[2021-02-05 19:18:08.589720] (SineGeneratorAgent_1): Pack time: 0.000765
[2021-02-05 19:18:08.592546] (MathAgent_1): Received: {'from': 'SineGeneratorAgent_1',
↪ 'data': array([0.]), 'senderType': 'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:18:08.595246] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.]), 'senderType': 'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:18:08.596230] (MultiMathAgent_1): Received: {'from':
↪'SineGeneratorAgent_1', 'data': array([0.]), 'senderType': 'SineGeneratorAgent',
↪'channel': 'default'}
[2021-02-05 19:18:08.595257] (SineGeneratorAgent_1): Sending: [0.]
[2021-02-05 19:18:08.593186] (MathAgent_1): Pack time: 0.000388
[2021-02-05 19:18:08.600139] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.])}
[2021-02-05 19:18:08.596854] (MultiMathAgent_1): Pack time: 0.000383
[2021-02-05 19:18:08.597959] (MultiMathAgent_1): Sending: {'minus': array([-0.5]),
↪'plus': array([0.5])}
[2021-02-05 19:18:08.593788] (MathAgent_1): Sending: [0.]
[2021-02-05 19:18:08.600521] (MonitorAgent_1): Tproc: 0.000931
[2021-02-05 19:18:08.598126] (MultiMathAgent_1): Tproc: 0.001735
[2021-02-05 19:18:08.594015] (MathAgent_1): Tproc: 0.001276
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:18:08.606396] (MonitorAgent_1): Received: {'from': 'MathAgent_1', 'data
↪': array([0.]), 'senderType': 'MathAgent', 'channel': 'default'}
[2021-02-05 19:18:08.607428] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.]), 'MathAgent_1': array([0.])}
[2021-02-05 19:18:08.607807] (MonitorAgent_1): Tproc: 0.001189
[2021-02-05 19:18:08.613784] (MonitorAgent_1): Received: {'from': 'MultiMathAgent_1',
↪'data': {'minus': array([-0.5]), 'plus': array([0.5])}, 'senderType':
↪'MultiMathAgent', 'channel': 'default'}
[2021-02-05 19:18:08.620942] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.]), 'MathAgent_1': array([0.]), 'MultiMathAgent_1': {'minus': array([-0.
↪5]), 'plus': array([0.5])}}
[2021-02-05 19:18:08.625194] (MonitorAgent_1): Tproc: 0.01115
[2021-02-05 19:18:09.590078] (SineGeneratorAgent_1): Pack time: 0.000561
[2021-02-05 19:18:09.592887] (SineGeneratorAgent_1): Sending: [0.06279052]
[2021-02-05 19:18:09.593616] (MathAgent_1): Received: {'from': 'SineGeneratorAgent_1',
↪ 'data': array([0.06279052]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:18:09.593134] (MultiMathAgent_1): Received: {'from':
↪'SineGeneratorAgent_1', 'data': array([0.06279052]), 'senderType':
↪'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:18:09.598873] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.06279052]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:18:09.608525] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052]), 'MathAgent_1': array([0.]), 'MultiMathAgent_1': {
↪'minus': array([-0.5]), 'plus': array([0.5])}}
[2021-02-05 19:18:09.594706] (MathAgent_1): Pack time: 0.000551
[2021-02-05 19:18:09.596935] (MultiMathAgent_1): Pack time: 0.001516
[2021-02-05 19:18:09.613332] (MonitorAgent_1): Tproc: 0.013659
[2021-02-05 19:18:09.596653] (MathAgent_1): Sending: [0.03139526]
[2021-02-05 19:18:09.613588] (MultiMathAgent_1): Sending: {'minus': array([-0.
↪43720948]), 'plus': array([0.56279052])}
[2021-02-05 19:18:09.620995] (MonitorAgent_1): Received: {'from': 'MathAgent_1', 'data
↪': array([0.03139526]), 'senderType': 'MathAgent', 'channel': 'default'}
[2021-02-05 19:18:09.597111] (MathAgent_1): Tproc: 0.003054
[2021-02-05 19:18:09.616401] (MultiMathAgent_1): Tproc: 0.022068
[2021-02-05 19:18:09.631221] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052]), 'MathAgent_1': array([0.        , 0.03139526]),
↪'MultiMathAgent_1': {'minus': array([-0.5]), 'plus': array([0.5])}}
[2021-02-05 19:18:09.632022] (MonitorAgent_1): Tproc: 0.0105
[2021-02-05 19:18:09.637249] (MonitorAgent_1): Received: {'from': 'MultiMathAgent_1',
↪'data': {'minus': array([-0.43720948]), 'plus': array([0.56279052])}, 'senderType':
↪'MultiMathAgent', 'channel': 'default'}
[2021-02-05 19:18:09.651789] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052]), 'MathAgent_1': array([0.        , 0.03139526]),
↪'MultiMathAgent_1': {'minus': array([-0.5       , -0.43720948]), 'plus': array([0.5␣
↪      , 0.56279052])}}
[2021-02-05 19:18:09.652908] (MonitorAgent_1): Tproc: 0.014285
```

6.3 Tutorial 3 - An advanced pipeline with multichannel signals.

We can use different channels for the receiver to handle specifically each channel name. This can be useful for example in splitting train and test channels in machine learning Then, the user will need to implement specific handling of each channel in the receiving agent.

In this example, the *MultiGeneratorAgent* is used to send two different types of data - Sine and Cosine generator. This is done via specifying `send_output (channel="sine")` and `send_output(channel="cosine")`.

Then on the receiving end, the `on_received_message()` function checks for `message['channel']` to handle it separately.

Note that by default, *MonitorAgent* is only subscribed to the `"default"` channel. Hence, it will not respond to the `"cosine"` and `"sine"` channel.

[2]:
```
# %load tutorial_3_multi_channel.py
from agentMET4FOF.agents import AgentMET4FOF, AgentNetwork, MonitorAgent
from agentMET4FOF.streams import SineGenerator, CosineGenerator


class MultiGeneratorAgent(AgentMET4FOF):

    _sine_stream: SineGenerator
    _cos_stream: CosineGenerator

    def init_parameters(self):
        self._sine_stream = SineGenerator()
        self._cos_stream = CosineGenerator()

    def agent_loop(self):
        if self.current_state == "Running":
            sine_data = self._sine_stream.next_sample()  # dictionary
            cosine_data = self._cos_stream.next_sample()  # dictionary
            self.send_output(sine_data["quantities"], channel="sine")
            self.send_output(cosine_data["quantities"], channel="cosine")


class MultiOutputMathAgent(AgentMET4FOF):

    _minus_param: float
    _plus_param: float

    def init_parameters(self, minus_param=0.5, plus_param=0.5):
        self._minus_param = minus_param
        self._plus_param = plus_param

    def on_received_message(self, message):
        """
        Checks for message['channel'] and handles them separately
        Acceptable channels are "cosine" and "sine"
        """
        if message["channel"] == "cosine":
            minus_data = self.minus(message["data"], self._minus_param)
            self.send_output({"cosine_minus": minus_data})
        elif message["channel"] == "sine":
            plus_data = self.plus(message["data"], self._plus_param)
            self.send_output({"sine_plus": plus_data})

    @staticmethod
    def minus(data, minus_val):
        return data - minus_val

    @staticmethod
    def plus(data, plus_val):
```

(continues on next page)

(continued from previous page)

```
        return data + plus_val


def main():
    # start agent network server
    agentNetwork = AgentNetwork()

    # init agents
    gen_agent = agentNetwork.add_agent(agentType=MultiGeneratorAgent)
    multi_math_agent = agentNetwork.add_agent(agentType=MultiOutputMathAgent)
    monitor_agent = agentNetwork.add_agent(agentType=MonitorAgent)

    # connect agents : We can connect multiple agents to any particular agent
    # However the agent needs to implement handling multiple inputs
    agentNetwork.bind_agents(gen_agent, multi_math_agent, channel=["sine","cosine"])
    agentNetwork.bind_agents(multi_math_agent, monitor_agent)

    # set all agents states to "Running"
    agentNetwork.set_running_state()

    # allow for shutting down the network after execution
    return agentNetwork


if __name__ == "__main__":
    main()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333
INFO [2021-07-01 15:01:52.175496] (MultiGeneratorAgent_1): INITIALIZED
INFO [2021-07-01 15:01:52.210891] (MultiOutputMathAgent_1): INITIALIZED
INFO [2021-07-01 15:01:52.249834] (MonitorAgent_1): INITIALIZED
[2021-07-01 15:01:52.267296] (MultiGeneratorAgent_1): Connected output module:␣
↪MultiOutputMathAgent_1
[2021-07-01 15:01:52.281811] (MultiOutputMathAgent_1): Connected output module:␣
↪MonitorAgent_1

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

SET STATE:   Running
[2021-07-01 15:01:53.182613] (MultiGeneratorAgent_1): Pack time: 0.001315
[2021-07-01 15:01:53.185913] (MultiGeneratorAgent_1): Sending: [0.]
[2021-07-01 15:01:53.188089] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.]), 'senderType': 'MultiGeneratorAgent',
↪'channel': 'sine'}
[2021-07-01 15:01:53.198228] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([0.5])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:53.187680] (MultiGeneratorAgent_1): Pack time: 0.000927
```

(continues on next page)

(continued from previous page)

```
[2021-07-01 15:01:53.190148] (MultiOutputMathAgent_1): Pack time: 0.001023
[2021-07-01 15:01:53.201389] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5])}}
[2021-07-01 15:01:53.191978] (MultiGeneratorAgent_1): Sending: [1.]
[2021-07-01 15:01:53.202160] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([0.5])}
[2021-07-01 15:01:53.205550] (MonitorAgent_1): Tproc: 0.006233
[2021-07-01 15:01:53.222512] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.5])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:53.203843] (MultiOutputMathAgent_1): Tproc: 0.014867
[2021-07-01 15:01:53.225894] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5]), 'cosine_minus': array([0.5])}}
[2021-07-01 15:01:53.206881] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([1.]), 'senderType': 'MultiGeneratorAgent',
↪'channel': 'cosine'}
[2021-07-01 15:01:53.226755] (MonitorAgent_1): Tproc: 0.003277
[2021-07-01 15:01:53.214200] (MultiOutputMathAgent_1): Pack time: 0.006701
[2021-07-01 15:01:53.216159] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.5])}
[2021-07-01 15:01:53.216622] (MultiOutputMathAgent_1): Tproc: 0.009249
[2021-07-01 15:01:54.184818] (MultiGeneratorAgent_1): Pack time: 0.000496
[2021-07-01 15:01:54.186555] (MultiGeneratorAgent_1): Sending: [0.58778525]
[2021-07-01 15:01:54.187543] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.58778525]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:01:54.191606] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([1.08778525])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:54.188158] (MultiGeneratorAgent_1): Pack time: 0.001101
[2021-07-01 15:01:54.188642] (MultiOutputMathAgent_1): Pack time: 0.000556
[2021-07-01 15:01:54.198646] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525]), 'cosine_minus': array([0.5])}}
[2021-07-01 15:01:54.192076] (MultiGeneratorAgent_1): Sending: [0.99802673]
[2021-07-01 15:01:54.193649] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([1.08778525])}
[2021-07-01 15:01:54.199203] (MonitorAgent_1): Tproc: 0.007071
[2021-07-01 15:01:54.206693] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.49802673])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:54.194338] (MultiOutputMathAgent_1): Tproc: 0.00634
[2021-07-01 15:01:54.211962] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525]), 'cosine_minus': array([0.5       , 0.
↪49802673])}}
[2021-07-01 15:01:54.198345] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.99802673]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:54.212776] (MonitorAgent_1): Tproc: 0.004846
[2021-07-01 15:01:54.201066] (MultiOutputMathAgent_1): Pack time: 0.001228
[2021-07-01 15:01:54.203439] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.49802673])}
[2021-07-01 15:01:54.204268] (MultiOutputMathAgent_1): Tproc: 0.004457
[2021-07-01 15:01:55.178989] (MultiGeneratorAgent_1): Pack time: 0.000235
[2021-07-01 15:01:55.179521] (MultiGeneratorAgent_1): Sending: [0.95105652]
[2021-07-01 15:01:55.179758] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.95105652]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
```

(continues on next page)

(continued from previous page)

```
[2021-07-01 15:01:55.180977] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([1.45105652])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:55.179716] (MultiGeneratorAgent_1): Pack time: 8.7e-05
[2021-07-01 15:01:55.179960] (MultiOutputMathAgent_1): Pack time: 9e-05
[2021-07-01 15:01:55.181787] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652]), 'cosine_minus': array([0.
↪5       , 0.49802673])}}
[2021-07-01 15:01:55.179963] (MultiGeneratorAgent_1): Sending: [0.9921147]
[2021-07-01 15:01:55.180249] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([1.45105652])}
[2021-07-01 15:01:55.181918] (MonitorAgent_1): Tproc: 0.000833
[2021-07-01 15:01:55.183711] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.4921147])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:55.180316] (MultiOutputMathAgent_1): Tproc: 0.000466
[2021-07-01 15:01:55.184531] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652]), 'cosine_minus': array([0.
↪5       , 0.49802673, 0.4921147 ])}}
[2021-07-01 15:01:55.181834] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.9921147]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:55.184744] (MonitorAgent_1): Tproc: 0.000897
[2021-07-01 15:01:55.182130] (MultiOutputMathAgent_1): Pack time: 0.000143
[2021-07-01 15:01:55.182865] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.4921147])}
[2021-07-01 15:01:55.183034] (MultiOutputMathAgent_1): Tproc: 0.001075
NS shut down.
[2021-07-01 15:01:56.180795] (MultiGeneratorAgent_1): Pack time: 0.000474
[2021-07-01 15:01:56.182502] (MultiGeneratorAgent_1): Sending: [0.95105652]
[2021-07-01 15:01:56.184165] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.95105652]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:01:56.188323] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([1.45105652])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:56.183937] (MultiGeneratorAgent_1): Pack time: 0.000939
[2021-07-01 15:01:56.186133] (MultiOutputMathAgent_1): Pack time: 0.001328
[2021-07-01 15:01:56.193199] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652]), 'cosine_minus
↪': array([0.5       , 0.49802673, 0.4921147 ])}}
[2021-07-01 15:01:56.186587] (MultiGeneratorAgent_1): Sending: [0.98228725]
[2021-07-01 15:01:56.188313] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([1.45105652])}
[2021-07-01 15:01:56.194104] (MonitorAgent_1): Tproc: 0.005235
[2021-07-01 15:01:56.197336] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.48228725])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:56.189350] (MultiOutputMathAgent_1): Tproc: 0.00465
[2021-07-01 15:01:56.202432] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652]), 'cosine_minus
↪': array([0.5       , 0.49802673, 0.4921147 , 0.48228725])}}
[2021-07-01 15:01:56.192299] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.98228725]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:56.203192] (MonitorAgent_1): Tproc: 0.005025
[2021-07-01 15:01:56.194134] (MultiOutputMathAgent_1): Pack time: 0.001021
```

(continues on next page)

(continued from previous page)

```
[2021-07-01 15:01:56.196129] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.48228725])}
[2021-07-01 15:01:56.196786] (MultiOutputMathAgent_1): Tproc: 0.003774
[2021-07-01 15:01:57.181941] (MultiGeneratorAgent_1): Pack time: 0.001002
[2021-07-01 15:01:57.187711] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.58778525]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:01:57.201793] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([1.08778525])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:57.184197] (MultiGeneratorAgent_1): Sending: [0.58778525]
[2021-07-01 15:01:57.193170] (MultiOutputMathAgent_1): Pack time: 0.004933
[2021-07-01 15:01:57.207510] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652, 1.08778525]),
↪'cosine_minus': array([0.5       , 0.49802673, 0.4921147 , 0.48228725])}}
[2021-07-01 15:01:57.193366] (MultiGeneratorAgent_1): Pack time: 0.008673
[2021-07-01 15:01:57.194990] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([1.08778525])}
[2021-07-01 15:01:57.208243] (MonitorAgent_1): Tproc: 0.005928
[2021-07-01 15:01:57.194990] (MultiGeneratorAgent_1): Sending: [0.96858316]
[2021-07-01 15:01:57.195499] (MultiOutputMathAgent_1): Tproc: 0.007365
[2021-07-01 15:01:57.213668] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.46858316])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:57.219609] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652, 1.08778525]),
↪'cosine_minus': array([0.5       , 0.49802673, 0.4921147 , 0.48228725, 0.46858316])}
↪}
[2021-07-01 15:01:57.199423] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.96858316]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:57.220649] (MonitorAgent_1): Tproc: 0.006413
[2021-07-01 15:01:57.202937] (MultiOutputMathAgent_1): Pack time: 0.002508
[2021-07-01 15:01:57.205253] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.46858316])}
[2021-07-01 15:01:57.206416] (MultiOutputMathAgent_1): Tproc: 0.006076
[2021-07-01 15:01:58.181290] (MultiGeneratorAgent_1): Pack time: 0.000689
[2021-07-01 15:01:58.183351] (MultiGeneratorAgent_1): Sending: [1.2246468e-16]
[2021-07-01 15:01:58.184399] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([1.2246468e-16]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:01:58.191755] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([0.5])}, 'senderType':
↪'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:58.187102] (MultiGeneratorAgent_1): Pack time: 0.002899
[2021-07-01 15:01:58.197160] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652, 1.08778525,
       0.5       ]), 'cosine_minus': array([0.5       , 0.49802673, 0.4921147 , 0.
↪48228725, 0.46858316])}}
[2021-07-01 15:01:58.188956] (MultiOutputMathAgent_1): Pack time: 0.000701
[2021-07-01 15:01:58.192717] (MultiGeneratorAgent_1): Sending: [0.95105652]
[2021-07-01 15:01:58.190516] (MultiOutputMathAgent_1): Sending: {'sine_plus':␣
↪array([0.5])}
[2021-07-01 15:01:58.197707] (MonitorAgent_1): Tproc: 0.004917
[2021-07-01 15:01:58.200727] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.45105652])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-07-01 15:01:58.190927] (MultiOutputMathAgent_1): Tproc: 0.002815
[2021-07-01 15:01:58.207440] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([0.5       , 1.08778525, 1.45105652, 1.45105652, 1.08778525,
       0.5       ]), 'cosine_minus': array([0.5       , 0.49802673, 0.4921147 , 0.
↪48228725, 0.46858316,
       0.45105652])}}
[2021-07-01 15:01:58.194263] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.95105652]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:58.208510] (MonitorAgent_1): Tproc: 0.007011
[2021-07-01 15:01:58.196419] (MultiOutputMathAgent_1): Pack time: 0.001563
[2021-07-01 15:01:58.198945] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.45105652])}
[2021-07-01 15:01:58.201557] (MultiOutputMathAgent_1): Tproc: 0.006677
[2021-07-01 15:01:59.180793] (MultiGeneratorAgent_1): Pack time: 0.000552
[2021-07-01 15:01:59.182349] (MultiGeneratorAgent_1): Sending: [-0.58778525]
[2021-07-01 15:01:59.183520] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([-0.58778525]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:01:59.190572] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([-0.08778525])}, 'senderType':
↪ 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:59.183271] (MultiGeneratorAgent_1): Pack time: 0.000478
[2021-07-01 15:01:59.184681] (MultiOutputMathAgent_1): Pack time: 0.000554
[2021-07-01 15:01:59.197807] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([ 0.5       ,  1.08778525,  1.45105652,  1.45105652,  1.08778525,
       0.5       , -0.08778525]), 'cosine_minus': array([0.5       , 0.49802673, 0.
↪4921147 , 0.48228725, 0.46858316,
       0.45105652])}}
[2021-07-01 15:01:59.185007] (MultiGeneratorAgent_1): Sending: [0.92977649]
[2021-07-01 15:01:59.186423] (MultiOutputMathAgent_1): Sending: {'sine_plus': array([-
↪0.08778525])}
[2021-07-01 15:01:59.198320] (MonitorAgent_1): Tproc: 0.007246
[2021-07-01 15:01:59.201333] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.42977649])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:01:59.186874] (MultiOutputMathAgent_1): Tproc: 0.002855
[2021-07-01 15:01:59.207784] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([ 0.5       ,  1.08778525,  1.45105652,  1.45105652,  1.08778525,
       0.5       , -0.08778525]), 'cosine_minus': array([0.5       , 0.49802673, 0.
↪4921147 , 0.48228725, 0.46858316,
       0.45105652, 0.42977649])}}
[2021-07-01 15:01:59.189887] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.92977649]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:01:59.208882] (MonitorAgent_1): Tproc: 0.006219
[2021-07-01 15:01:59.192005] (MultiOutputMathAgent_1): Pack time: 0.001011
[2021-07-01 15:01:59.193943] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.42977649])}
[2021-07-01 15:01:59.194633] (MultiOutputMathAgent_1): Tproc: 0.003875
[2021-07-01 15:02:00.181478] (MultiGeneratorAgent_1): Pack time: 0.000776
[2021-07-01 15:02:00.184191] (MultiGeneratorAgent_1): Sending: [-0.95105652]
[2021-07-01 15:02:00.197991] (MultiGeneratorAgent_1): Pack time: 0.013129
[2021-07-01 15:02:00.187946] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([-0.95105652]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'sine'}
[2021-07-01 15:02:00.207903] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'sine_plus': array([-0.45105652])}, 'senderType':
↪ 'MultiOutputMathAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-07-01 15:02:00.205156] (MultiGeneratorAgent_1): Sending: [0.90482705]
[2021-07-01 15:02:00.190476] (MultiOutputMathAgent_1): Pack time: 0.000718
[2021-07-01 15:02:00.229208] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([ 0.5       ,  1.08778525,  1.45105652,  1.45105652,  1.08778525,
       0.5       , -0.08778525, -0.45105652]), 'cosine_minus': array([0.5       , 0.
↪49802673, 0.4921147 , 0.48228725, 0.46858316,
      0.45105652, 0.42977649])}}
[2021-07-01 15:02:00.229840] (MonitorAgent_1): Tproc: 0.021469
[2021-07-01 15:02:00.196944] (MultiOutputMathAgent_1): Sending: {'sine_plus': array([-
↪0.45105652])}
[2021-07-01 15:02:00.232820] (MonitorAgent_1): Received: {'from':
↪'MultiOutputMathAgent_1', 'data': {'cosine_minus': array([0.40482705])}, 'senderType
↪': 'MultiOutputMathAgent', 'channel': 'default'}
[2021-07-01 15:02:00.201511] (MultiOutputMathAgent_1): Tproc: 0.011616
[2021-07-01 15:02:00.241196] (MonitorAgent_1): Buffer: {'MultiOutputMathAgent_1': {
↪'sine_plus': array([ 0.5       ,  1.08778525,  1.45105652,  1.45105652,  1.08778525,
       0.5       , -0.08778525, -0.45105652]), 'cosine_minus': array([0.5       , 0.
↪49802673, 0.4921147 , 0.48228725, 0.46858316,
      0.45105652, 0.42977649, 0.40482705])}}
[2021-07-01 15:02:00.221224] (MultiOutputMathAgent_1): Received: {'from':
↪'MultiGeneratorAgent_1', 'data': array([0.90482705]), 'senderType':
↪'MultiGeneratorAgent', 'channel': 'cosine'}
[2021-07-01 15:02:00.241435] (MonitorAgent_1): Tproc: 0.007897
[2021-07-01 15:02:00.223049] (MultiOutputMathAgent_1): Pack time: 0.000661
[2021-07-01 15:02:00.224864] (MultiOutputMathAgent_1): Sending: {'cosine_minus':␣
↪array([0.40482705])}
[2021-07-01 15:02:00.225436] (MultiOutputMathAgent_1): Tproc: 0.003688
```

6.4 Tutorial 4 - A metrological datastream

In this tutorial we introduce the new metrologically enabled agents. We initialize an agent, which generates an infinite sine signal. The signal is generated from the built-in class `MetrologicalSineGenerator` which delivers on each call one timestamp and one value each with associated uncertainties.

The *MetrologicalSineGeneratorAgent* is based on the new class *agentMET4FOF.metrological_agents.MetrologicalAgent*. We only adapt the methods `init_parameters()` and `agent_loop()`. This we need to hand over an instance of the signal generating class and to generate the actual samples. The rest of the buffering and plotting logic is encapsulated inside of the new base classes.

[2]:
```
# %load tutorial_4_metrological_streams.py
from agentMET4FOF.agents import AgentNetwork
from agentMET4FOF.metrological_agents import MetrologicalAgent,␣
↪MetrologicalMonitorAgent
from agentMET4FOF.metrological_streams import (
    MetrologicalDataStreamMET4FOF,
    MetrologicalSineGenerator,
)


class MetrologicalSineGeneratorAgent(MetrologicalAgent):
    """An agent streaming a sine signal

    Takes samples from an instance of :py:class:`MetrologicalSineGenerator` and pushes
    them sample by sample to connected agents via its output channel.
```

(continues on next page)

(continued from previous page)

```
    """

    # The datatype of the stream will be MetrologicalSineGenerator.
    _stream: MetrologicalDataStreamMET4FOF

    def init_parameters(
        self,
        signal: MetrologicalDataStreamMET4FOF = MetrologicalSineGenerator(),
        **kwargs
    ):
        """Initialize the input data stream

        Parameters
        ----------
        signal : MetrologicalDataStreamMET4FOF
            the underlying signal for the generator
        """
        self._stream = signal
        super().init_parameters()
        self.set_output_data(channel="default", metadata=self._stream.metadata)

    def agent_loop(self):
        """Model the agent's behaviour

        On state *Running* the agent will extract sample by sample the input
        datastream's content and push it into its output buffer.
        """
        if self.current_state == "Running":
            self.set_output_data(channel="default", data=[self._stream.next_sample()])
            super().agent_loop()


def demonstrate_metrological_stream():

    # start agent network server
    agent_network = AgentNetwork(dashboard_modules=True)

    # Initialize signal generating class outside of agent framework.
    signal = MetrologicalSineGenerator()

    # Initialize metrologically enabled agent taking name from signal source metadata.
    source_name = signal.metadata.metadata["device_id"]
    source_agent = agent_network.add_agent(
        name=source_name, agentType=MetrologicalSineGeneratorAgent
    )
    source_agent.init_parameters(signal)

    # Initialize metrologically enabled plotting agent.
    monitor_agent = agent_network.add_agent(
        "MonitorAgent",
        agentType=MetrologicalMonitorAgent,
        buffer_size=50,
    )

    # Bind agents.
    source_agent.bind_output(monitor_agent)
```

(continues on next page)

(continued from previous page)

```
    # Set all agents states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution.
    return agent_network


if __name__ == "__main__":
    demonstrate_metrological_stream()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-02-05 19:24:37.900901] (SineGenerator): INITIALIZED
INFO [2021-02-05 19:24:37.939928] (MonitorAgent): INITIALIZED
[2021-02-05 19:24:37.955210] (SineGenerator): Connected output module: MonitorAgent
SET STATE:   Running
[2021-02-05 19:24:38.907664] (SineGenerator): Pack time: 0.000336
[2021-02-05 19:24:38.909019] (SineGenerator): Sending: [array([[0.        , 0.
↪, 0.68494649, 0.25      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e871730>]
[2021-02-05 19:24:38.910202] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.        , 0.        , 0.68494649, 0.25      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7f424e822bb0>, 'senderType':
↪'MetrologicalSineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:24:38.911237] (MonitorAgent): Buffer: {'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.68494649, 0.25      ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7f424e822bb0>]}}
[2021-02-05 19:24:38.911459] (MonitorAgent): Tproc: 0.000968
[2021-02-05 19:24:39.907438] (SineGenerator): Pack time: 0.000403
[2021-02-05 19:24:39.909067] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.002     , 0.        , 0.42028269, 0.25      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7f424e822c70>, 'senderType':
↪'MetrologicalSineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:24:39.908668] (SineGenerator): Sending: [array([[0.002     , 0.
↪, 0.42028269, 0.25      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e871730>]
[2021-02-05 19:24:39.910652] (MonitorAgent): Buffer: {'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.68494649, 0.25      ],
       [0.002     , 0.        , 0.42028269, 0.25      ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7f424e822bb0>, <time_series_metadata.scheme.
↪MetaData object at 0x7f424e822c70>]}}
[2021-02-05 19:24:39.911060] (MonitorAgent): Tproc: 0.001731
[2021-02-05 19:24:40.908816] (SineGenerator): Pack time: 0.000738
[2021-02-05 19:24:40.912960] (SineGenerator): Sending: [array([[0.004     , 0.
↪, 1.54415656, 0.25      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e871730>]
[2021-02-05 19:24:40.913508] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.004     , 0.        , 1.54415656, 0.25      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7f424e822ca0>, 'senderType':
↪'MetrologicalSineGeneratorAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:24:40.917793] (MonitorAgent): Buffer: {'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.68494649, 0.25      ],
       [0.002     , 0.        , 0.42028269, 0.25      ],
       [0.004     , 0.        , 1.54415656, 0.25      ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7f424e822bb0>, <time_series_metadata.scheme.
↪MetaData object at 0x7f424e822c70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e822ca0>]}}
[2021-02-05 19:24:40.919608] (MonitorAgent): Tproc: 0.005357
[2021-02-05 19:24:41.907632] (SineGenerator): Pack time: 0.000437
[2021-02-05 19:24:41.909056] (SineGenerator): Sending: [array([[0.006    , 0.       ,␣
↪0.9256774, 0.25     ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e871730>]
[2021-02-05 19:24:41.909673] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.006    , 0.       , 0.9256774, 0.25     ]]), 'metadata': <time_series_
↪metadata.scheme.MetaData object at 0x7f424e822d00>, 'senderType':
↪'MetrologicalSineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:24:41.913785] (MonitorAgent): Buffer: {'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.68494649, 0.25      ],
       [0.002     , 0.        , 0.42028269, 0.25      ],
       [0.004     , 0.        , 1.54415656, 0.25      ],
       [0.006     , 0.        , 0.9256774 , 0.25      ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7f424e822bb0>, <time_series_metadata.scheme.
↪MetaData object at 0x7f424e822c70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e822ca0>, <time_series_metadata.scheme.MetaData object at 0x7f424e822d00>]}}
[2021-02-05 19:24:41.914125] (MonitorAgent): Tproc: 0.004137
[2021-02-05 19:24:42.909007] (SineGenerator): Pack time: 0.000784
[2021-02-05 19:24:42.911438] (SineGenerator): Sending: [array([[0.008     , 0.       ␣
↪, 1.12170929, 0.25      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e871730>]
[2021-02-05 19:24:42.913822] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.008     , 0.        , 1.12170929, 0.25      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7f424e822490>, 'senderType':
↪'MetrologicalSineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:24:42.920323] (MonitorAgent): Buffer: {'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.68494649, 0.25      ],
       [0.002     , 0.        , 0.42028269, 0.25      ],
       [0.004     , 0.        , 1.54415656, 0.25      ],
       [0.006     , 0.        , 0.9256774 , 0.25      ],
       [0.008     , 0.        , 1.12170929, 0.25      ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7f424e822bb0>, <time_series_metadata.scheme.
↪MetaData object at 0x7f424e822c70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7f424e822ca0>, <time_series_metadata.scheme.MetaData object at 0x7f424e822d00>,
↪<time_series_metadata.scheme.MetaData object at 0x7f424e822490>]}}
[2021-02-05 19:24:42.921078] (MonitorAgent): Tproc: 0.006613
```

6.5 Tutorial 5 - Building coalitions

We demonstrate the use of Coalition of agents to group agents together. Rationale of grouping depends on the users and application. For example, we can group sensors which are measuring the same measurand. To this end, the coalition consists of a list of agent names and provides aesthetic differences in the dashboard.

All coalitions visible by the `agent_network` can be accessed via `agent_network.coalitions`.

[1]:
```
# %load tutorial_5_coalition.py
from agentMET4FOF.agents import AgentNetwork, MonitorAgent, SineGeneratorAgent


def demonstrate_generator_agent_use():
    # Start agent network server.
    agent_network = AgentNetwork()

    # Initialize agents by adding them to the agent network.
    gen_agent_1 = agent_network.add_agent(agentType=SineGeneratorAgent)
    gen_agent_2 = agent_network.add_agent(agentType=SineGeneratorAgent)
    monitor_agent = agent_network.add_agent(agentType=MonitorAgent)

    # bind generator agents outputs to monitor
    agent_network.bind_agents(gen_agent_1, monitor_agent)
    agent_network.bind_agents(gen_agent_2, monitor_agent)

    # setup health coalition group
    agent_network.add_coalition(
        "REDUNDANT_SENSORS", [gen_agent_1, gen_agent_2, monitor_agent]
    )

    # Set all agents' states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution
    return agent_network


if __name__ == "__main__":
    demonstrate_generator_agent_use()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-02-05 19:34:50.491737] (SineGeneratorAgent_1): INITIALIZED
INFO [2021-02-05 19:34:50.524928] (SineGeneratorAgent_2): INITIALIZED
INFO [2021-02-05 19:34:50.557538] (MonitorAgent_1): INITIALIZED
[2021-02-05 19:34:50.574726] (SineGeneratorAgent_1): Connected output module:␣
↪MonitorAgent_1
[2021-02-05 19:34:50.583164] (SineGeneratorAgent_2): Connected output module:␣
↪MonitorAgent_1
SET STATE:   Running
[2021-02-05 19:34:51.499491] (SineGeneratorAgent_1): Pack time: 0.001512
[2021-02-05 19:34:51.502797] (SineGeneratorAgent_1): Sending: [0.]
[2021-02-05 19:34:51.507439] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.]), 'senderType': 'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:34:51.543076] (SineGeneratorAgent_2): Pack time: 0.002763
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:34:51.509952] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.])}
[2021-02-05 19:34:51.546875] (SineGeneratorAgent_2): Sending: [0.]
[2021-02-05 19:34:51.511171] (MonitorAgent_1): Tproc: 0.002834
[2021-02-05 19:34:51.544997] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.]), 'senderType': 'SineGeneratorAgent', 'channel': 'default'}
[2021-02-05 19:34:51.554497] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.]), 'SineGeneratorAgent_2': array([0.])}
[2021-02-05 19:34:51.555370] (MonitorAgent_1): Tproc: 0.009853
[2021-02-05 19:34:52.495894] (SineGeneratorAgent_1): Pack time: 0.00018
[2021-02-05 19:34:52.496930] (SineGeneratorAgent_1): Sending: [0.06279052]
[2021-02-05 19:34:52.497336] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.06279052]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:52.498583] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052]), 'SineGeneratorAgent_2': array([0.])}
[2021-02-05 19:34:52.498796] (MonitorAgent_1): Tproc: 0.001265
[2021-02-05 19:34:52.528407] (SineGeneratorAgent_2): Pack time: 0.000181
[2021-02-05 19:34:52.529454] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.06279052]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:52.529242] (SineGeneratorAgent_2): Sending: [0.06279052]
[2021-02-05 19:34:52.530661] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052]), 'SineGeneratorAgent_2': array([0.        , 0.
↪06279052])}
[2021-02-05 19:34:52.530827] (MonitorAgent_1): Tproc: 0.001183
[2021-02-05 19:34:53.496614] (SineGeneratorAgent_1): Pack time: 0.000331
[2021-02-05 19:34:53.497841] (SineGeneratorAgent_1): Sending: [0.12533323]
[2021-02-05 19:34:53.499000] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.12533323]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:53.535469] (SineGeneratorAgent_2): Pack time: 0.000494
[2021-02-05 19:34:53.503180] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323]), 'SineGeneratorAgent_2': array([0.      ␣
↪  , 0.06279052])}
[2021-02-05 19:34:53.537256] (SineGeneratorAgent_2): Sending: [0.12533323]
[2021-02-05 19:34:53.503676] (MonitorAgent_1): Tproc: 0.004153
[2021-02-05 19:34:53.541731] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.12533323]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:53.545969] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323]), 'SineGeneratorAgent_2': array([0.      ␣
↪  , 0.06279052, 0.12533323])}
[2021-02-05 19:34:53.554543] (MonitorAgent_1): Tproc: 0.011906
NS shut down.
[2021-02-05 19:34:54.497690] (SineGeneratorAgent_1): Pack time: 0.000662
[2021-02-05 19:34:54.504170] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.18738131]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:54.502583] (SineGeneratorAgent_1): Sending: [0.18738131]
[2021-02-05 19:34:54.515421] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131]), 'SineGeneratorAgent_2':␣
↪array([0.        , 0.06279052, 0.12533323])}
[2021-02-05 19:34:54.528529] (SineGeneratorAgent_2): Pack time: 0.000237
[2021-02-05 19:34:54.529196] (SineGeneratorAgent_2): Sending: [0.18738131]
[2021-02-05 19:34:54.519541] (MonitorAgent_1): Tproc: 0.014505
[2021-02-05 19:34:54.531254] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.18738131]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:34:54.532868] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131]), 'SineGeneratorAgent_2':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131])}
[2021-02-05 19:34:54.533451] (MonitorAgent_1): Tproc: 0.001834
[2021-02-05 19:34:55.497332] (SineGeneratorAgent_1): Pack time: 0.00057
[2021-02-05 19:34:55.499010] (SineGeneratorAgent_1): Sending: [0.24868989]
[2021-02-05 19:34:55.500133] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.24868989]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:55.533920] (SineGeneratorAgent_2): Pack time: 0.001052
[2021-02-05 19:34:55.538313] (SineGeneratorAgent_2): Sending: [0.24868989]
[2021-02-05 19:34:55.503852] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989]),
↪'SineGeneratorAgent_2': array([0.        , 0.06279052, 0.12533323, 0.18738131])}
[2021-02-05 19:34:55.504382] (MonitorAgent_1): Tproc: 0.003753
[2021-02-05 19:34:55.538319] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.24868989]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:55.544134] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989]),
↪'SineGeneratorAgent_2': array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.
↪24868989])}
[2021-02-05 19:34:55.545007] (MonitorAgent_1): Tproc: 0.005636
[2021-02-05 19:34:56.496439] (SineGeneratorAgent_1): Pack time: 0.000345
[2021-02-05 19:34:56.499405] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.30901699]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:56.497380] (SineGeneratorAgent_1): Sending: [0.30901699]
[2021-02-05 19:34:56.503408] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699]), 'SineGeneratorAgent_2': array([0.        , 0.06279052, 0.
↪12533323, 0.18738131, 0.24868989])}
[2021-02-05 19:34:56.530998] (SineGeneratorAgent_2): Pack time: 0.000148
[2021-02-05 19:34:56.503616] (MonitorAgent_1): Tproc: 0.003954
[2021-02-05 19:34:56.531354] (SineGeneratorAgent_2): Sending: [0.30901699]
[2021-02-05 19:34:56.532137] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.30901699]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:56.532882] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699]), 'SineGeneratorAgent_2': array([0.        , 0.06279052, 0.
↪12533323, 0.18738131, 0.24868989,
       0.30901699])}
[2021-02-05 19:34:56.533007] (MonitorAgent_1): Tproc: 0.000751
[2021-02-05 19:34:57.497217] (SineGeneratorAgent_1): Pack time: 0.000607
[2021-02-05 19:34:57.498990] (SineGeneratorAgent_1): Sending: [0.36812455]
[2021-02-05 19:34:57.499886] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.36812455]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:57.529841] (SineGeneratorAgent_2): Pack time: 0.000718
[2021-02-05 19:34:57.505595] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455]), 'SineGeneratorAgent_2': array([0.        , 0.
↪06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699])}
[2021-02-05 19:34:57.531642] (SineGeneratorAgent_2): Sending: [0.36812455]
[2021-02-05 19:34:57.506484] (MonitorAgent_1): Tproc: 0.006025
```

(continues on next page)

(continued from previous page)

```
[2021-02-05 19:34:57.548550] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.36812455]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:57.568741] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455]), 'SineGeneratorAgent_2': array([0.        , 0.
↪06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455])}
[2021-02-05 19:34:57.570422] (MonitorAgent_1): Tproc: 0.021339
[2021-02-05 19:34:58.495766] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪1', 'data': array([0.42577929]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:58.496176] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929]), 'SineGeneratorAgent_2': array([0.
↪, 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455])}
[2021-02-05 19:34:58.495362] (SineGeneratorAgent_1): Pack time: 7.8e-05
[2021-02-05 19:34:58.496226] (MonitorAgent_1): Tproc: 0.000388
[2021-02-05 19:34:58.495629] (SineGeneratorAgent_1): Sending: [0.42577929]
[2021-02-05 19:34:58.527965] (SineGeneratorAgent_2): Pack time: 9.5e-05
[2021-02-05 19:34:58.528730] (MonitorAgent_1): Received: {'from': 'SineGeneratorAgent_
↪2', 'data': array([0.42577929]), 'senderType': 'SineGeneratorAgent', 'channel':
↪'default'}
[2021-02-05 19:34:58.528211] (SineGeneratorAgent_2): Sending: [0.42577929]
[2021-02-05 19:34:58.529372] (MonitorAgent_1): Buffer: {'SineGeneratorAgent_1':␣
↪array([0.        , 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929]), 'SineGeneratorAgent_2': array([0.
↪, 0.06279052, 0.12533323, 0.18738131, 0.24868989,
       0.30901699, 0.36812455, 0.42577929])}
[2021-02-05 19:34:58.529423] (MonitorAgent_1): Tproc: 0.000622
```

6.6 Tutorial 6 - Using a different backend

By default, "osbrain" backend offers real connectivity between agents (each agent has its own port & IP address) in distributed systems (e,g connecting agents from raspberry pis to PCs, etc), which explains why it is harder to debug.

In the "mesa" backend, there's only one real timer which is started in the AgentNetwork, and every timer tick will advance the agent actions by calling `step()` which includes `agent_loop` and `on_received_message`. Moreover, in the "mesa" backend, agents do not have their own port and IP addresses, they are simulated objects to emulate the behaviour of distributed agents. Hence, "osbrain" is closer to deployment phase, whereas mesa is suited for the simulation/designing phase. To switch between the backends, simply pass the backend parameter to either "mesa" or "osbrain" in the AgentNetwork instantiation.

[2]:
```
# %load tutorial_6_mesa_backend.py
from agentMET4FOF.agents import AgentNetwork, MonitorAgent, SineGeneratorAgent


def demonstrate_mesa_backend():

    # Start agent network and specify backend via the corresponding keyword parameter.
    _agent_network = AgentNetwork(backend="mesa")

    # Initialize agents by adding them to the agent network.
```

(continues on next page)

(continued from previous page)

```
    sine_agent = _agent_network.add_agent(agentType=SineGeneratorAgent)
    monitor_agent = _agent_network.add_agent(agentType=MonitorAgent, buffer_size=200)
    sine_agent.bind_output(monitor_agent)

    # Set all agents states to "Running".
    _agent_network.set_running_state()

    return _agent_network


if __name__ == "__main__":
    demonstrate_mesa_backend()
```

```
------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------
[2021-02-05 19:39:41.143529] (SineGeneratorAgent_1): INITIALIZED

[2021-02-05 19:39:41.143892] (MonitorAgent_1): INITIALIZED
[2021-02-05 19:39:41.143941] (SineGeneratorAgent_1): Connected output module:␣
↪MonitorAgent_1
SET STATE:   Running
```

6.7 Tutorial 7 - Generating signals using a generic metrological agent

In the present tutorial we demonstrate the creation of arbitrary metrologically enabled agents that generate signals according to a given function. agent network with three metrologically enabled agents, two of which are defined as objects of the `MetrologicalGeneratorAgent` class and the third is a single monitor agent that simultaneously plots the output of the first two agents. The first agent takes an input from an object of the `MetrologicalSineGenerator` class (ref. Tutorial 4) and the second from a `MetrologicalMultiwaveGenerator` object, which generates a signal given by a sum of cosines.

The *MetrologicalGeneratorAgent* is based on the *agentMET4FOF.metrological_agents.MetrologicalAgent* class.

[3]:
```
# %load tutorial_7_generic_metrological_agent.py
from agentMET4FOF.agents import AgentNetwork
from agentMET4FOF.metrological_agents import (
    MetrologicalMonitorAgent,
    MetrologicalGeneratorAgent,
)
from agentMET4FOF.metrological_streams import (
    MetrologicalSineGenerator,
    MetrologicalMultiWaveGenerator,
)


def demonstrate_metrological_stream():
    """Demonstrate an agent network with two metrologically enabled agents
```

(continues on next page)

(continued from previous page)

```
    The agents are defined as objects of the :class:`MetrologicalGeneratorAgent`
    class whose outputs are bound to a single monitor agent.

    The metrological agents generate signals from a sine wave and a multiwave generator
    source.

    Returns
    -------
    :class:`AgentNetwork`
        The initialized and running agent network object
    """
    # start agent network server
    agent_network = AgentNetwork(dashboard_modules=True)

    # Initialize metrologically enabled agent with a multiwave (sum of cosines)
    # generator as signal source taking name from signal source metadata.
    signal_multiwave = MetrologicalMultiWaveGenerator(
        quantity_names="Voltage", quantity_units="V"
    )
    source_name_multiwave = signal_multiwave.metadata.metadata["device_id"]
    source_agent_multiwave = agent_network.add_agent(
        name=source_name_multiwave, agentType=MetrologicalGeneratorAgent
    )
    source_agent_multiwave.init_parameters(signal=signal_multiwave)

    # Initialize second metrologically enabled agent with a sine generator as signal
    # source taking name from signal source metadata.
    signal_sine = MetrologicalSineGenerator()
    source_name_sine = signal_sine.metadata.metadata["device_id"]
    source_agent_sine = agent_network.add_agent(
        name=source_name_sine, agentType=MetrologicalGeneratorAgent
    )
    source_agent_sine.init_parameters(signal=signal_sine)

    # Initialize metrologically enabled plotting agent.
    monitor_agent = agent_network.add_agent(
        "MonitorAgent",
        agentType=MetrologicalMonitorAgent,
        buffer_size=50,
    )

    # Bind agents.
    source_agent_multiwave.bind_output(monitor_agent)
    source_agent_sine.bind_output(monitor_agent)

    # Set all agents states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution.
    return agent_network


if __name__ == "__main__":
    demonstrate_metrological_stream()
```

```
Starting NameServer...
```

(continues on next page)

(continued from previous page)

```
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-06-04 08:55:23.204159] (MultiWaveDataGenerator): INITIALIZED
INFO [2021-06-04 08:55:23.239519] (SineGenerator): INITIALIZED
INFO [2021-06-04 08:55:23.264925] (MonitorAgent): INITIALIZED
[2021-06-04 08:55:23.281027] (MultiWaveDataGenerator): Connected output module:␣
↪MonitorAgent
[2021-06-04 08:55:23.287882] (SineGenerator): Connected output module: MonitorAgent
SET STATE:   Running
/home/ludwig10/code/agentMET4FOF/agentMET4FOF/metrological_streams.py:135:␣
↪UserWarning:

No uncertainty generator function specified. Setting to default (constant).

[2021-06-04 08:55:24.215457] (MultiWaveDataGenerator): Pack time: 0.000883
[2021-06-04 08:55:24.221676] (MultiWaveDataGenerator): Sending: [array([[0.        ,␣
↪0.        , 1.14201382, 0.1       ]]), <time_series_metadata.scheme.MetaData object␣
↪at 0x7fcb3440d3a0>]
[2021-06-04 08:55:24.224453] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[0.        , 0.        , 1.14201382, 0.1  ␣
↪    ]]), 'metadata': <time_series_metadata.scheme.MetaData object at 0x7fcb34407790>
↪, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:24.229078] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>]}}
[2021-06-04 08:55:24.230706] (MonitorAgent): Tproc: 0.00426
[2021-06-04 08:55:24.244909] (SineGenerator): Pack time: 0.000908
[2021-06-04 08:55:24.246596] (SineGenerator): Sending: [array([[0.        , 0.       ␣
↪, 0.28402764, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:24.246392] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.        , 0.        , 0.28402764, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34412e20>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:24.247713] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>]}, 'SineGenerator': {'data
↪': array([[0.        , 0.        , 0.28402764, 0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34412e20>]}}
[2021-06-04 08:55:24.247870] (MonitorAgent): Tproc: 0.001326
[2021-06-04 08:55:25.208567] (MultiWaveDataGenerator): Pack time: 0.000196
[2021-06-04 08:55:25.209381] (MultiWaveDataGenerator): Sending: [array([[0.002     ,␣
↪0.        , 0.85706774, 0.1       ]]), <time_series_metadata.scheme.MetaData object␣
↪at 0x7fcb3440d3a0>]
[2021-06-04 08:55:25.209775] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[0.002     , 0.        , 0.85706774, 0.1  ␣
↪    ]]), 'metadata': <time_series_metadata.scheme.MetaData object at 0x7fcb34412520>
↪, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-06-04 08:55:25.211461] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ],
       [0.002     , 0.        , 0.85706774, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412520>]}, 'SineGenerator': {'data': array([[0.        ,␣
↪0.        , 0.28402764, 0.1       ]]), 'metadata': [<time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412e20>]}}
[2021-06-04 08:55:25.211678] (MonitorAgent): Tproc: 0.001699
[2021-06-04 08:55:25.242644] (SineGenerator): Pack time: 0.000117
[2021-06-04 08:55:25.243075] (SineGenerator): Sending: [array([[0.002     , 0.       ␣
↪, 0.68388675, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:25.243264] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.002     , 0.        , 0.68388675, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34412f40>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:25.244287] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ],
       [0.002     , 0.        , 0.85706774, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412520>]}, 'SineGenerator': {'data': array([[0.        ,␣
↪0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>]}}
[2021-06-04 08:55:25.244407] (MonitorAgent): Tproc: 0.001012
NS shut down.
[2021-06-04 08:55:26.211862] (MultiWaveDataGenerator): Pack time: 0.000771
[2021-06-04 08:55:26.214571] (MultiWaveDataGenerator): Sending: [array([[0.004     ,␣
↪0.        , 0.36951548, 0.1       ]]), <time_series_metadata.scheme.MetaData object␣
↪at 0x7fcb3440d3a0>]
[2021-06-04 08:55:26.215381] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[0.004     , 0.        , 0.36951548, 0.1␣
↪    ]]), 'metadata': <time_series_metadata.scheme.MetaData object at 0x7fcb34402160>
↪, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:26.246395] (SineGenerator): Pack time: 0.001739
[2021-06-04 08:55:26.226046] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ],
       [0.002     , 0.        , 0.85706774, 0.1       ],
       [0.004     , 0.        , 0.36951548, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402160>]}, 'SineGenerator': {'data': array([[0.        , 0.        , 0.
↪28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>]}}
[2021-06-04 08:55:26.255335] (SineGenerator): Sending: [array([[0.004     , 0.       ␣
↪, 1.07205349, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:26.226903] (MonitorAgent): Tproc: 0.007832
[2021-06-04 08:55:26.258551] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.004     , 0.        , 1.07205349, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34402af0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:26.267521] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[0.        , 0.        , 1.14201382, 0.1       ],
```

(continues on next page)

(continued from previous page)

```
       [0.002     , 0.        , 0.85706774, 0.1       ],
       [0.004     , 0.        , 0.36951548, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402160>]}, 'SineGenerator': {'data': array([[0.        , 0.        , 0.
↪28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>]}}
[2021-06-04 08:55:26.268465] (MonitorAgent): Tproc: 0.00931
[2021-06-04 08:55:27.211643] (MultiWaveDataGenerator): Pack time: 0.000733
[2021-06-04 08:55:27.214634] (MultiWaveDataGenerator): Sending: [array([[ 0.006     ,␣
↪ 0.        , -0.23075114,  0.1       ]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb3440d3a0>]
[2021-06-04 08:55:27.215842] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[ 0.006     ,  0.        , -0.23075114,  0.
↪1       ]]), 'metadata': <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:27.245239] (SineGenerator): Pack time: 0.000761
[2021-06-04 08:55:27.248372] (SineGenerator): Sending: [array([[0.006     , 0.       ␣
↪, 1.10758822, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:27.232388] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>]}, 'SineGenerator': {'data': array([[0.        , 0.        , 0.
↪28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>]}}
[2021-06-04 08:55:27.236935] (MonitorAgent): Tproc: 0.020195
[2021-06-04 08:55:27.254223] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.006     , 0.        , 1.10758822, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34402ac0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:27.277781] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>]}, 'SineGenerator': {'data': array([[0.        , 0.        , 0.
↪28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>]}}
```

(continues on next page)

(continued from previous page)

```
[2021-06-04 08:55:27.278651] (MonitorAgent): Tproc: 0.023881
[2021-06-04 08:55:28.211305] (MultiWaveDataGenerator): Pack time: 0.000477
[2021-06-04 08:55:28.212788] (MultiWaveDataGenerator): Sending: [array([[ 0.008     ,␣
↪ 0.        , -0.83424568,  0.1       ]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb3440d3a0>]
[2021-06-04 08:55:28.214757] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[ 0.008     ,  0.        , -0.83424568,  0.
↪1       ]]), 'metadata': <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb344029d0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:28.220546] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>]},
↪'SineGenerator': {'data': array([[0.        , 0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>]}}
[2021-06-04 08:55:28.221139] (MonitorAgent): Tproc: 0.006031
[2021-06-04 08:55:28.243869] (SineGenerator): Pack time: 0.000181
[2021-06-04 08:55:28.244381] (SineGenerator): Sending: [array([[0.008     , 0.       ␣
↪, 0.53732789, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:28.246182] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.008     , 0.        , 0.53732789, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34402a90>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:28.248224] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>]},
↪'SineGenerator': {'data': array([[0.        , 0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ],
       [0.008     , 0.        , 0.53732789, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb34402a90>]}}
[2021-06-04 08:55:28.248419] (MonitorAgent): Tproc: 0.002012
[2021-06-04 08:55:29.210769] (MultiWaveDataGenerator): Pack time: 0.00045
[2021-06-04 08:55:29.214621] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[ 0.01      ,  0.        , -0.99847908,  1.
↪1       ]]), 'metadata': <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb344029a0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-06-04 08:55:29.211959] (MultiWaveDataGenerator): Sending: [array([[ 0.01      ,␣
↪ 0.        , -0.99847908,  0.1       ]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb3440d3a0>]
[2021-06-04 08:55:29.221246] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ],
       [ 0.01      ,  0.        , -0.99847908,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb344029a0>]}, 'SineGenerator':
↪{'data': array([[0.        , 0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ],
       [0.008     , 0.        , 0.53732789, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb34402a90>]}}
[2021-06-04 08:55:29.221505] (MonitorAgent): Tproc: 0.006553
[2021-06-04 08:55:29.242710] (SineGenerator): Pack time: 0.000168
[2021-06-04 08:55:29.243189] (SineGenerator): Sending: [array([[0.01      , 0.       ␣
↪, 0.00304184, 0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:29.244172] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[0.01      , 0.        , 0.00304184, 0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407a00>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:29.246427] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ],
       [ 0.01      ,  0.        , -0.99847908,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb344029a0>]}, 'SineGenerator':
↪{'data': array([[0.        , 0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ],
       [0.008     , 0.        , 0.53732789, 0.1       ],
       [0.01      , 0.        , 0.00304184, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb34402a90>, <time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407a00>]}}
[2021-06-04 08:55:29.246649] (MonitorAgent): Tproc: 0.002336
```

(continues on next page)

(continued from previous page)

```
[2021-06-04 08:55:30.208630] (MultiWaveDataGenerator): Pack time: 0.000272
[2021-06-04 08:55:30.209391] (MultiWaveDataGenerator): Sending: [array([[ 0.012     ,␣
↪ 0.        , -0.82945933,  0.1       ]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb3440d3a0>]
[2021-06-04 08:55:30.209873] (MonitorAgent): Received: {'from':
↪'MultiWaveDataGenerator', 'data': array([[ 0.012     ,  0.        , -0.82945933,  0.
↪1       ]]), 'metadata': <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402a60>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:30.212460] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ],
       [ 0.01      ,  0.        , -0.99847908,  0.1       ],
       [ 0.012     ,  0.        , -0.82945933,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb344029a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fcb34402a60>]}, 'SineGenerator': {'data':␣
↪array([[0.        , 0.        , 0.28402764, 0.1       ],
       [0.002     , 0.        , 0.68388675, 0.1       ],
       [0.004     , 0.        , 1.07205349, 0.1       ],
       [0.006     , 0.        , 1.10758822, 0.1       ],
       [0.008     , 0.        , 0.53732789, 0.1       ],
       [0.01      , 0.        , 0.00304184, 0.1       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402ac0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb34402a90>, <time_series_
↪metadata.scheme.MetaData object at 0x7fcb34407a00>]}}
[2021-06-04 08:55:30.212723] (MonitorAgent): Tproc: 0.002663
[2021-06-04 08:55:30.242541] (SineGenerator): Pack time: 0.000137
[2021-06-04 08:55:30.242887] (SineGenerator): Sending: [array([[ 0.012     ,  0.      ␣
↪  , -0.62866992,  0.1       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34352730>]
[2021-06-04 08:55:30.243528] (MonitorAgent): Received: {'from': 'SineGenerator', 'data
↪': array([[ 0.012     ,  0.        , -0.62866992,  0.1       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fcb34402a00>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-06-04 08:55:30.244778] (MonitorAgent): Buffer: {'MultiWaveDataGenerator': {'data
↪': array([[ 0.        ,  0.        ,  1.14201382,  0.1       ],
       [ 0.002     ,  0.        ,  0.85706774,  0.1       ],
       [ 0.004     ,  0.        ,  0.36951548,  0.1       ],
       [ 0.006     ,  0.        , -0.23075114,  0.1       ],
       [ 0.008     ,  0.        , -0.83424568,  0.1       ],
       [ 0.01      ,  0.        , -0.99847908,  0.1       ],
       [ 0.012     ,  0.        , -0.82945933,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34407790>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412520>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402160>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402b50>, <time_series_metadata.scheme.MetaData object at 0x7fcb344029d0>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb344029a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fcb34402a60>]}, 'SineGenerator': {'data':␣
↪array([[ 0.        ,  0.        ,  0.28402764,  0.1       ],
```

(continues on next page)

(continued from previous page)

```
       [ 0.002     ,  0.        ,  0.68388675,  0.1       ],
       [ 0.004     ,  0.        ,  1.07205349,  0.1       ],
       [ 0.006     ,  0.        ,  1.10758822,  0.1       ],
       [ 0.008     ,  0.        ,  0.53732789,  0.1       ],
       [ 0.01      ,  0.        ,  0.00304184,  0.1       ],
       [ 0.012     ,  0.        , -0.62866992,  0.1       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fcb34412e20>, <time_series_metadata.
↪scheme.MetaData object at 0x7fcb34412f40>, <time_series_metadata.scheme.MetaData␣
↪object at 0x7fcb34402af0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fcb34402ac0>, <time_series_metadata.scheme.MetaData object at 0x7fcb34402a90>,
↪<time_series_metadata.scheme.MetaData object at 0x7fcb34407a00>, <time_series_
↪metadata.scheme.MetaData object at 0x7fcb34402a00>]}}
[2021-06-04 08:55:30.244881] (MonitorAgent): Tproc: 0.001262
```

CHAPTER 7

Working with signals carrying redundant information

7.1 Redundancy Agent – Determining redundancy in several similar signals

In this tutorial we generate four signals and supply the data to a Redundancy Agent. The agent calculates the best consistent estimate taking into account the supplied uncertainties. It rejects sensor values that may be erroneous. In this presented use case the sensors directly measure the measurand.

At the start of the main module we define all important parameters. For detailed descriptions of the parameters please refer to the docstrings of the respective methods. Then we define the agents and start the network. The network and the calculated results can be monitored in a browser at the address http://127.0.0.1:8050/.

[3]:
```
# %load redundancy_agent_four_signals.py
import numpy as np

from agentMET4FOF.agents.metrological_base_agents import (
    MetrologicalMonitorAgent,
)
from agentMET4FOF.agents.metrological_redundancy_agents import RedundancyAgent
from agentMET4FOF.agents.metrological_signal_agents import (
    MetrologicalGeneratorAgent,
)
from agentMET4FOF.network import AgentNetwork
from agentMET4FOF.streams.metrological_signal_streams import (
    MetrologicalMultiWaveGenerator,
)


def demonstrate_redundancy_agent_four_signals():
    batch_size = 10
    n_pr = batch_size
    fsam = 100
    intercept = 10
```

(continues on next page)

(continued from previous page)

```
    frequencies = [6, 10, 8, 12]
    phases = [1, 2, 3, 4]
    amplitudes = [0.3, 0.2, 0.5, 0.4]
    exp_unc_abs = 0.2
    probability_limit = 0.95

    # start agent network server
    agent_network: AgentNetwork = AgentNetwork(dashboard_modules=True)

    # Initialize signal generating class outside of agent framework.
    signal_arr = [
        MetrologicalMultiWaveGenerator(
            sfreq=fsam,
            freq_arr=np.array([frequency]),
            ampl_arr=np.array([amplitude]),
            phase_ini_arr=np.array([phase]),
            intercept=intercept,
            value_unc=exp_unc_abs,
        )
        for frequency, amplitude, phase in zip(frequencies, amplitudes, phases)
    ]

    # Data source agents.
    source_agents = []
    sensor_key_list = []
    for count, signal in enumerate(signal_arr):
        sensor_key_list += ["Sensor" + str(count + 1)]
        source_agents += [
            agent_network.add_agent(
                name=sensor_key_list[-1], agentType=MetrologicalGeneratorAgent
            )
        ]
        source_agents[-1].init_parameters(signal=signal, batch_size=batch_size)

    # Redundant data processing agent
    redundancy_name1 = "RedundancyAgent1"
    redundancy_agent1 = agent_network.add_agent(
        name=redundancy_name1, agentType=RedundancyAgent
    )
    redundancy_agent1.init_parameters(
        sensor_key_list=sensor_key_list,
        n_pr=n_pr,
        problim=probability_limit,
        calc_type="lcs",
    )

    # Initialize metrologically enabled plotting agent.
    monitor_agent1 = agent_network.add_agent(
        name="MonitorAgent_SensorValues", agentType=MetrologicalMonitorAgent
    )
    monitor_agent2 = agent_network.add_agent(
        name="MonitorAgent_RedundantEstimate", agentType=MetrologicalMonitorAgent
    )

    # Bind agents.
    for source_agent in source_agents:
        source_agent.bind_output(monitor_agent1)
```

(continues on next page)

(continued from previous page)

```
        source_agent.bind_output(redundancy_agent1)

    redundancy_agent1.bind_output(monitor_agent2)

    # Set all agents states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution.
    return agent_network


if __name__ == "__main__":
    demonstrate_redundancy_agent_four_signals()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333
INFO [2021-07-07 14:49:18.993883] (Sensor1): INITIALIZED
INFO [2021-07-07 14:49:19.035303] (Sensor2): INITIALIZED
INFO [2021-07-07 14:49:19.072919] (Sensor3): INITIALIZED
INFO [2021-07-07 14:49:19.110387] (Sensor4): INITIALIZED
INFO [2021-07-07 14:49:19.143509] (RedundancyAgent1): INITIALIZED
```

```
/home/ludwig10/code/agentMET4FOF/agentMET4FOF/metrological_streams.py:135:␣
↪UserWarning:

No uncertainty generator function specified. Setting to default (value_unc = constant,
↪ time_unc = 0).
```

```
------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-07-07 14:49:19.188230] (MonitorAgent_SensorValues): INITIALIZED
INFO [2021-07-07 14:49:19.220150] (MonitorAgent_RedundantEstimate): INITIALIZED
[2021-07-07 14:49:19.245339] (Sensor1): Connected output module: MonitorAgent_
↪SensorValues
[2021-07-07 14:49:19.253119] (Sensor1): Connected output module: RedundancyAgent1
[2021-07-07 14:49:19.261596] (Sensor2): Connected output module: MonitorAgent_
↪SensorValues
[2021-07-07 14:49:19.269916] (Sensor2): Connected output module: RedundancyAgent1
[2021-07-07 14:49:19.278184] (Sensor3): Connected output module: MonitorAgent_
↪SensorValues
[2021-07-07 14:49:19.292819] (Sensor3): Connected output module: RedundancyAgent1
[2021-07-07 14:49:19.302562] (Sensor4): Connected output module: MonitorAgent_
↪SensorValues
[2021-07-07 14:49:19.311015] (Sensor4): Connected output module: RedundancyAgent1
[2021-07-07 14:49:19.327098] (RedundancyAgent1): Connected output module:␣
↪MonitorAgent_RedundantEstimate
SET STATE:   Running
[2021-07-07 14:49:20.009515] (Sensor1): Pack time: 0.003875
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:20.011230] (Sensor1): Sending: [array([[ 0.        ,  0.        ,
↪10.05758677,  0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad47d550>]
[2021-07-07 14:49:20.013703] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5e20>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.014299] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.015177] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>]}}
[2021-07-07 14:49:20.014521] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪  ,  0.        , 10.05758677,  0.2       ]])}
[2021-07-07 14:49:20.015372] (MonitorAgent_SensorValues): Tproc: 0.000836
[2021-07-07 14:49:20.014668] (RedundancyAgent1): Tproc: 0.000764
[2021-07-07 14:49:20.044422] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.044547] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5e50>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.042673] (Sensor2): Pack time: 0.000218
[2021-07-07 14:49:20.045087] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>]}, 'Sensor2': {'data':
↪array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>]}}
[2021-07-07 14:49:20.045181] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪  ,  0.        , 10.05758677,  0.2       ]]), 'Sensor2': array([[0.        , 0.
↪  , 9.81226671, 0.2       ]])}
[2021-07-07 14:49:20.043540] (Sensor2): Sending: [array([[0.        , 0.        , 9.
↪81226671, 0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad4806d0>]
[2021-07-07 14:49:20.046343] (MonitorAgent_SensorValues): Tproc: 0.001796
[2021-07-07 14:49:20.045304] (RedundancyAgent1): Tproc: 0.000627
[2021-07-07 14:49:20.078476] (Sensor3): Pack time: 0.000241
[2021-07-07 14:49:20.079205] (Sensor3): Sending: [array([[0.        , 0.        , 9.
↪40049983, 0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad4826d0>]
[2021-07-07 14:49:20.080191] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[0.        , 0.        , 9.40049983, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5f40>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.081131] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[0.        , 0.        , 9.40049983, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.082066] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>]}, 'Sensor2': {'data':
↪array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>]}, 'Sensor3': {'data':
↪array([[0.        , 0.        , 9.40049983, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3cea30>]}}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:20.081235] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪  ,  0.        , 10.05758677,  0.2       ]]), 'Sensor2': array([[0.        , 0.
↪   , 9.81226671, 0.2       ]]), 'Sensor3': array([[0.        , 0.        , 9.
↪40049983, 0.2       ]])}
[2021-07-07 14:49:20.082157] (MonitorAgent_SensorValues): Tproc: 0.000925
[2021-07-07 14:49:20.081328] (RedundancyAgent1): Tproc: 0.00101
[2021-07-07 14:49:20.116046] (Sensor4): Pack time: 0.000222
[2021-07-07 14:49:20.116813] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[0.        , 0.        , 9.63403863, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.116792] (Sensor4): Sending: [array([[0.        , 0.        , 9.
↪63403863, 0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad482700>]
[2021-07-07 14:49:20.117888] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.        ,  0.        , 10.05758677,  0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>]}, 'Sensor2': {'data':
↪array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>]}, 'Sensor3': {'data':
↪array([[0.        , 0.        , 9.40049983, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3cea30>]}, 'Sensor4': {'data':
↪array([[0.        , 0.        , 9.63403863, 0.2       ]]), 'metadata': [<time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce730>]}}
[2021-07-07 14:49:20.120198] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[0.        , 0.        , 9.63403863, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:20.121533] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪  ,  0.        , 10.05758677,  0.2       ]]), 'Sensor2': array([[0.        , 0.
↪   , 9.81226671, 0.2       ]]), 'Sensor3': array([[0.        , 0.        , 9.
↪40049983, 0.2       ]]), 'Sensor4': array([[0.        , 0.        , 9.63403863, 0.2
↪      ]])}
[2021-07-07 14:49:20.117995] (MonitorAgent_SensorValues): Tproc: 0.001085
[2021-07-07 14:49:20.121671] (RedundancyAgent1): Tproc: 0.001357
Buffer size is  1 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:21.007073] (Sensor1): Pack time: 0.005616
[2021-07-07 14:49:21.010574] (Sensor1): Sending: [array([[0.01      , 0.        , 9.
↪85659121, 0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad47d550>]
[2021-07-07 14:49:21.015850] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.01      , 0.        , 9.85659121, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3cea00>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.015851] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.01      , 0.        , 9.85659121, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.050802] (Sensor2): Pack time: 0.004355
[2021-07-07 14:49:21.078180] (Sensor3): Pack time: 0.000268
[2021-07-07 14:49:21.078841] (Sensor3): Sending: [array([[0.01      , 0.        , 9.
↪33105188, 0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad4826d0>]
[2021-07-07 14:49:21.062134] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>]}, 'Sensor2': {'data': array([[0.
↪        , 0.        , 9.81226671, 0.2       ]]), 'metadata': [<time_series_metadata.
↪scheme.MetaData object at 0x7fecad3ce7f0>]}, 'Sensor3': {'data': array([[0.
↪0.        , 9.40049983, 0.2       ]]), 'metadata': [<time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cea30>]}, 'Sensor4': {'data': array([[0.        , 0.
↪    , 9.63403863, 0.2       ]]), 'metadata': [<time_series_metadata.scheme.MetaData
↪object at 0x7fecad3ce730>]}}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:21.030692] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ]]), 'Sensor3': array([[0.   ␣
↪    , 0.        , 9.40049983, 0.2       ]]), 'Sensor4': array([[0.        , 0.     ␣
↪  , 9.63403863, 0.2       ]])}
[2021-07-07 14:49:21.059748] (Sensor2): Sending: [array([[0.01      , 0.        , 9.
↪62458464, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:21.031615] (RedundancyAgent1): Tproc: 0.01496
[2021-07-07 14:49:21.066776] (MonitorAgent_SensorValues): Tproc: 0.050095
[2021-07-07 14:49:21.053804] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5e80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.068216] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ceaf0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.067974] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'Sensor3': array([[0.   ␣
↪  , 0.        , 9.40049983, 0.2       ]]), 'Sensor4': array([[0.        , 0.     ␣
↪, 9.63403863, 0.2       ]])}
[2021-07-07 14:49:21.075919] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>]}, 'Sensor2': {'data': array([[0.
↪        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>]}, 'Sensor3': {'data': array([[0.        , 0.   ␣
↪    , 9.40049983, 0.2       ]]), 'metadata': [<time_series_metadata.scheme.MetaData␣
↪object at 0x7fecad3cea30>]}, 'Sensor4': {'data': array([[0.        , 0.        , 9.
↪63403863, 0.2       ]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7fecad3ce730>]}}
[2021-07-07 14:49:21.068204] (RedundancyAgent1): Tproc: 0.008848
[2021-07-07 14:49:21.076217] (MonitorAgent_SensorValues): Tproc: 0.003266
[2021-07-07 14:49:21.115605] (Sensor4): Pack time: 0.000165
[2021-07-07 14:49:21.081931] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5f70>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.081709] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ce700>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.116036] (Sensor4): Sending: [array([[0.01      , 0.        , 9.
↪81544544, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:21.084608] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
```

(continues on next page)

(continued from previous page)

```
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'Sensor3': array([[0.
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'Sensor4': array([[0.
↪  , 0.        , 9.63403863, 0.2       ]])}
[2021-07-07 14:49:21.085668] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>]}, 'Sensor2': {'data': array([[0.
↪        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>]}, 'Sensor3': {'data': array([[0.        , 0.
↪    , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>]}, 'Sensor4': {'data': array([[0.        , 0.
↪    , 9.63403863, 0.2       ]]), 'metadata': [<time_series_metadata.scheme.MetaData
↪object at 0x7fecad3ce730>]}}
[2021-07-07 14:49:21.084815] (RedundancyAgent1): Tproc: 0.002653
[2021-07-07 14:49:21.085823] (MonitorAgent_SensorValues): Tproc: 0.003889
[2021-07-07 14:49:21.119944] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.116997] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ceb20>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:21.121792] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'Sensor2':
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'Sensor3': array([[0.
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'Sensor4': array([[0.
↪  , 0.        , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]])}
[2021-07-07 14:49:21.118684] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>]}, 'Sensor2': {'data': array([[0.
↪        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>]}, 'Sensor3': {'data': array([[0.        , 0.
↪    , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>]}, 'Sensor4': {'data': array([[0.        , 0.
↪    , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb20>]}}
Buffer size is [2021-07-07 14:49:21.121989] (RedundancyAgent1): Tproc: 0.001862
[2021-07-07 14:49:21.118792] (MonitorAgent_SensorValues): Tproc: 0.001691
```

(continues on next page)

(continued from previous page)

```
 2 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:22.000958] (Sensor1): Pack time: 0.000504
[2021-07-07 14:49:22.003373] (Sensor1): Sending: [array([[0.02    , 0.      , 9.
↪999462, 0.2     ]]), <time_series_metadata.scheme.MetaData object at 0x7fecad47d550>
↪]
[2021-07-07 14:49:22.004863] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.02    , 0.      , 9.999462, 0.2     ]]), 'metadata': <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.005235] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.02    , 0.      , 9.999462, 0.2     ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3ce8b0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.019157] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>]}, 'Sensor2': {'data': array([[0.        , 0.
↪    , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>]}, 'Sensor3': {'data': array([[0.        , 0.
↪    , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>]}, 'Sensor4': {'data': array([[0.        , 0.
↪    , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb20>]}}
[2021-07-07 14:49:22.017290] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ]]), 'Sensor3': array([[0.
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'Sensor4': array([[0.
↪  , 0.        , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]])}
[2021-07-07 14:49:22.019977] (MonitorAgent_SensorValues): Tproc: 0.014061
[2021-07-07 14:49:22.018444] (RedundancyAgent1): Tproc: 0.01276
[2021-07-07 14:49:22.060122] (Sensor2): Pack time: 0.011586
[2021-07-07 14:49:22.082246] (Sensor3): Pack time: 0.001515
[2021-07-07 14:49:22.103090] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3cebb0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.052865] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5f70>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.062683] (Sensor2): Sending: [array([[0.02      , 0.        , 9.
↪85543299, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:22.086023] (Sensor3): Sending: [array([[0.02      , 0.        , 9.
↪72930289, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
[2021-07-07 14:49:22.119263] (Sensor4): Pack time: 0.000268
[2021-07-07 14:49:22.121752] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>]}, 'Sensor2': {'data': array([[0.        , 0.    ␣
↪    , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cebb0>]}, 'Sensor3': {'data': array([[0.        , 0.        , 9.40049983,␣
↪0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>]}, 'Sensor4': {'data': array([[0.        , 0.    ␣
↪    , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb20>]}}
[2021-07-07 14:49:22.116567] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'Sensor3': array([[0.     ␣
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ]]), 'Sensor4': array([[0.     ␣
↪  , 0.        , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]])}
[2021-07-07 14:49:22.120321] (Sensor4): Sending: [array([[ 0.02      ,  0.        ,␣
↪10.33981753,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:22.123789] (MonitorAgent_SensorValues): Tproc: 0.019983
[2021-07-07 14:49:22.117188] (RedundancyAgent1): Tproc: 0.062733
[2021-07-07 14:49:22.128020] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3cec10>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.118922] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5e80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.133617] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>]}, 'Sensor2': {'data': array([[0.        , 0.    ␣
↪    , 9.81226671, 0.2       ],
```

(continues on next page)

(continued from previous page)

```
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cebb0>]}, 'Sensor3': {'data': array([[0.        , 0.        , 9.40049983,␣
↪0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cec10>]}, 'Sensor4': {'data': array([[0.        , 0.        , 9.63403863,␣
↪0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb20>]}}
Buffer size is [2021-07-07 14:49:22.125997] (RedundancyAgent1): Buffer: {'Sensor1':␣
↪array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'Sensor3': array([[0.      ␣
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'Sensor4': array([[0.      ␣
↪  , 0.        , 9.63403863, 0.2       ],
       [0.01      , 0.        , 9.81544544, 0.2       ]])}
[2021-07-07 14:49:22.141930] (MonitorAgent_SensorValues): Tproc: 0.013113
 [2021-07-07 14:49:22.130103] (RedundancyAgent1): Tproc: 0.01089
[2021-07-07 14:49:22.142661] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[ 0.02      ,  0.        , 10.33981753,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3cec70>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
3[2021-07-07 14:49:22.136847] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data
↪': array([[ 0.02      ,  0.        , 10.33981753,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:22.153547] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>]}, 'Sensor2': {'data': array([[0.        , 0.    ␣
↪    , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cebb0>]}, 'Sensor3': {'data': array([[0.        , 0.        , 9.40049983,␣
↪0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cec10>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>]}}
 [2021-07-07 14:49:22.148906] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'Sensor3': array([[0.     ␣
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'Sensor4': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]])}
, which is less than  [2021-07-07 14:49:22.153718] (MonitorAgent_SensorValues): Tproc:
↪ 0.006877
10[2021-07-07 14:49:22.149517] (RedundancyAgent1): Tproc: 0.012471
 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:23.001249] (Sensor1): Pack time: 0.000518
[2021-07-07 14:49:23.003768] (Sensor1): Sending: [array([[ 0.03      ,  0.        ,␣
↪10.08057762,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:23.004714] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[ 0.03      ,  0.        , 10.08057762,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.004796] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[ 0.03      ,  0.        , 10.08057762,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3cea90>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.024964] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>]}, 'Sensor2': {'data': array([[0.        , 0.        , 9.81226671,␣
↪0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cebb0>]}, 'Sensor3': {'data': array([[0.        , 0.        , 9.40049983,␣
↪0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cec10>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>]}}
[2021-07-07 14:49:23.024965] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.        , 0.        , 9.81226671, 0.2       ],
       [0.01      , 0.        , 9.62458464, 0.2       ],
       [0.02      , 0.        , 9.85543299, 0.2       ]]), 'Sensor3': array([[0.      ␣
↪  , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'Sensor4': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]])}
[2021-07-07 14:49:23.047632] (Sensor2): Pack time: 0.001273
[2021-07-07 14:49:23.025808] (MonitorAgent_SensorValues): Tproc: 0.020348
[2021-07-07 14:49:23.025682] (RedundancyAgent1): Tproc: 0.020082
[2021-07-07 14:49:23.061407] (Sensor2): Sending: [array([[ 0.03      ,  0.        ,␣
↪10.09273912,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:23.078518] (Sensor3): Pack time: 0.000409
[2021-07-07 14:49:23.054290] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[ 0.03      ,  0.        , 10.09273912,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ced30>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.090960] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[ 0.03      ,  0.        , 10.09273912,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5e80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.079809] (Sensor3): Sending: [array([[ 0.03      ,  0.        ,␣
↪10.13847663,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
[2021-07-07 14:49:23.096215] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>]}, 'Sensor3': {'data': array([[0.        , 0.        , 9.40049983,␣
↪0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'metadata': [<time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cec10>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>]}}
[2021-07-07 14:49:23.098862] (MonitorAgent_SensorValues): Tproc: 0.036086
[2021-07-07 14:49:23.100533] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.03      ,  0.        , 10.13847663,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ced90>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.109361] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.        , 0.        , 9.40049983, 0.2       ],
       [0.01      , 0.        , 9.33105188, 0.2       ],
       [0.02      , 0.        , 9.72930289, 0.2       ]]), 'Sensor4': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]])}
[2021-07-07 14:49:23.110163] (RedundancyAgent1): Tproc: 0.01711
[2021-07-07 14:49:23.112103] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>]}}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:23.115326] (Sensor4): Pack time: 0.000146
[2021-07-07 14:49:23.112999] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[ 0.03      ,  0.        , 10.13847663,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5f70>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.115634] (MonitorAgent_SensorValues): Tproc: 0.012721
[2021-07-07 14:49:23.115816] (Sensor4): Sending: [array([[ 0.03      ,  0.        ,␣
↪10.63988826,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:23.122295] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01]])}
[2021-07-07 14:49:23.116724] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[ 0.03      ,  0.        , 10.63988826,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3cedf0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.122456] (RedundancyAgent1): Tproc: 0.008684
[2021-07-07 14:49:23.124353] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>]}}
[2021-07-07 14:49:23.124676] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[ 0.03      ,  0.        , 10.63988826,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3d5ee0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:23.124532] (MonitorAgent_SensorValues): Tproc: 0.00655
[2021-07-07 14:49:23.129010] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]])}
[2021-07-07 14:49:23.129574] (RedundancyAgent1): Tproc: 0.004748
Buffer size is  4 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:24.000868] (Sensor1): Pack time: 0.000566
[2021-07-07 14:49:24.003067] (Sensor1): Sending: [array([[0.04      , 0.        , 9.
↪66887351, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:24.004152] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.04      , 0.        , 9.66887351, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.005978] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.04      , 0.        , 9.66887351, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.023807] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>]},
↪'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>]}}
[2021-07-07 14:49:24.031224] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]])}
[2021-07-07 14:49:24.024723] (MonitorAgent_SensorValues): Tproc: 0.018109
[2021-07-07 14:49:24.031762] (RedundancyAgent1): Tproc: 0.027094
[2021-07-07 14:49:24.053222] (Sensor2): Pack time: 0.008118
NS shut down.
[2021-07-07 14:49:24.055885] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[0.04      , 0.        , 9.87107856, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459c70>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.050573] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[0.04      , 0.        , 9.87107856, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.055786] (Sensor2): Sending: [array([[0.04      , 0.        , 9.
↪87107856, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:24.080184] (Sensor3): Pack time: 0.000415
[2021-07-07 14:49:24.094510] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>]},
↪'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>]},
↪'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>]}}
[2021-07-07 14:49:24.081616] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]])}
[2021-07-07 14:49:24.115317] (Sensor4): Pack time: 0.000148
[2021-07-07 14:49:24.081397] (Sensor3): Sending: [array([[ 0.04      ,  0.        ,␣
↪10.05755346,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:24.096476] (MonitorAgent_SensorValues): Tproc: 0.039947
[2021-07-07 14:49:24.082143] (RedundancyAgent1): Tproc: 0.030634
[2021-07-07 14:49:24.115746] (Sensor4): Sending: [array([[ 0.04      ,  0.        ,␣
↪10.20797569,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:24.097377] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.04      ,  0.        , 10.05755346,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.086882] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[ 0.04      ,  0.        , 10.05755346,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.101512] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>]},
↪'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>]},
↪'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>]},
↪'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>]}}
[2021-07-07 14:49:24.094842] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01]])}
[2021-07-07 14:49:24.101812] (MonitorAgent_SensorValues): Tproc: 0.004319
[2021-07-07 14:49:24.095046] (RedundancyAgent1): Tproc: 0.007816
Buffer size is  [2021-07-07 14:49:24.116662] (MonitorAgent_SensorValues): Received: {
↪'from': 'Sensor4', 'data': array([[ 0.04      ,  0.        , 10.20797569,  0.2       ␣
↪ ]]), 'metadata': <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.121756] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[ 0.04      ,  0.        , 10.20797569,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:24.119812] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>]},
↪'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>]},
↪'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>]},
↪'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.
↪00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>]}}
[2021-07-07 14:49:24.126948] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]])}
[2021-07-07 14:49:24.119949] (MonitorAgent_SensorValues): Tproc: 0.003144
[2021-07-07 14:49:24.127060] (RedundancyAgent1): Tproc: 0.005193
5 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:25.000773] (Sensor1): Pack time: 0.000549
[2021-07-07 14:49:25.003259] (Sensor1): Sending: [array([[0.05      , 0.        , 9.
↪71880656, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:25.004381] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.05      , 0.        , 9.71880656, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:25.004406] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.05      , 0.        , 9.71880656, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:25.045974] (Sensor2): Pack time: 0.000723
[2021-07-07 14:49:25.027310] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>]}, 'Sensor2': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>]},
↪'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>]},
↪'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>]}}
[2021-07-07 14:49:25.034306] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]])}
[2021-07-07 14:49:25.081564] (Sensor3): Pack time: 0.001522
[2021-07-07 14:49:25.048315] (Sensor2): Sending: [array([[ 0.05      ,  0.        ,␣
↪10.09221065,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:25.028520] (MonitorAgent_SensorValues): Tproc: 0.023271
[2021-07-07 14:49:25.035852] (RedundancyAgent1): Tproc: 0.03091
[2021-07-07 14:49:25.084255] (Sensor3): Sending: [array([[ 0.05      ,  0.        ,␣
↪10.36796753,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:25.116279] (Sensor4): Pack time: 0.000292
[2021-07-07 14:49:25.117125] (Sensor4): Sending: [array([[ 0.05      ,  0.        ,
↪10.04256986,  0.2       ]]), <time_series_metadata.scheme.MetaData object at
↪0x7fecad482700>]
[2021-07-07 14:49:25.058309] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[ 0.05      ,  0.        , 10.09221065,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:25.052796] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[ 0.05      ,  0.        , 10.09221065,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:25.077909] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'Sensor2':
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'Sensor3':
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'Sensor4':
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]])}
[2021-07-07 14:49:25.101770] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>]}, 'Sensor2': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>]}, 'Sensor3': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>]},
↪'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.
↪00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>]}}
Buffer size is [2021-07-07 14:49:25.078517] (RedundancyAgent1): Tproc: 0.025156
 [2021-07-07 14:49:25.102277] (MonitorAgent_SensorValues): Tproc: 0.043267
6[2021-07-07 14:49:25.093159] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data
↪': array([[ 0.05      ,  0.        , 10.36796753,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:25.105473] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.05      ,  0.        , 10.36796753,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad459970>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
 , which is less than  10[2021-07-07 14:49:25.118593] (RedundancyAgent1): Buffer: {
↪'Sensor1': array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]])}
[2021-07-07 14:49:25.125922] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>]}, 'Sensor2': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>]}, 'Sensor3': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>]}, 'Sensor4': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>]}}
 .[2021-07-07 14:49:25.118823] (RedundancyAgent1): Tproc: 0.025122

[2021-07-07 14:49:25.127248] (MonitorAgent_SensorValues): Tproc: 0.021354
Not enough data for redundancy agent evaluation.[2021-07-07 14:49:25.119880]␣
↪(RedundancyAgent1): Received: {'from': 'Sensor4', 'data': array([[ 0.05      ,  0. ␣
↪      , 10.04256986,  0.2       ]]), 'metadata': <time_series_metadata.scheme.
↪MetaData object at 0x7fecad46b1c0>, 'senderType': 'MetrologicalGeneratorAgent',
↪'channel': 'default'}
[2021-07-07 14:49:25.128182] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[ 0.05      ,  0.        , 10.04256986,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}

[2021-07-07 14:49:25.128032] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]])}
[2021-07-07 14:49:25.133657] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>]}, 'Sensor2': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>]}, 'Sensor3': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>]}, 'Sensor4': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>]}}
[2021-07-07 14:49:25.128168] (RedundancyAgent1): Tproc: 0.008096
[2021-07-07 14:49:25.133884] (MonitorAgent_SensorValues): Tproc: 0.00555
[2021-07-07 14:49:25.998685] (Sensor1): Pack time: 0.000196
[2021-07-07 14:49:25.999185] (Sensor1): Sending: [array([[0.06      , 0.        , 9.
↪70455528, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:26.000340] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.06      , 0.        , 9.70455528, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ce4c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.000331] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.06      , 0.        , 9.70455528, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.004294] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]])}
[2021-07-07 14:49:26.004583] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>]}, 'Sensor2': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>]}, 'Sensor3': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>]}, 'Sensor4': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>]}}
[2021-07-07 14:49:26.004392] (RedundancyAgent1): Tproc: 0.00391
[2021-07-07 14:49:26.004709] (MonitorAgent_SensorValues): Tproc: 0.004226
[2021-07-07 14:49:26.041176] (Sensor2): Pack time: 0.000134
[2021-07-07 14:49:26.041835] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[ 0.06      ,  0.        , 10.17661344,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ce670>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.041503] (Sensor2): Sending: [array([[ 0.06      ,  0.        ,␣
↪10.17661344,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:26.046772] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[ 0.06      ,  0.        , 10.17661344,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.045984] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad41fe80>, <
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>]}, 'Sensor2': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01]
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>]}, 'Sensor3': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>]}, 'Sensor4': {'data
↪': array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>]}}
[2021-07-07 14:49:26.048356] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]])}
[2021-07-07 14:49:26.046153] (MonitorAgent_SensorValues): Tproc: 0.00423
[2021-07-07 14:49:26.048471] (RedundancyAgent1): Tproc: 0.001597
[2021-07-07 14:49:26.076987] (Sensor3): Pack time: 0.00014
[2021-07-07 14:49:26.077908] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[ 0.06      ,  0.        , 10.48463468,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.077584] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.06      ,  0.        , 10.48463468,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ce8e0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.077343] (Sensor3): Sending: [array([[ 0.06      ,  0.        ,␣
↪10.48463468,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
[2021-07-07 14:49:26.080354] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]])}
[2021-07-07 14:49:26.080207] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad41fe80>, <
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>]}, 'Sensor2': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01]
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>]}, 'Sensor3': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>]}, 'Sensor4': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>]}}
[2021-07-07 14:49:26.080303] (MonitorAgent_SensorValues): Tproc: 0.002634
[2021-07-07 14:49:26.080417] (RedundancyAgent1): Tproc: 0.002434
[2021-07-07 14:49:26.115261] (Sensor4): Pack time: 0.000242
[2021-07-07 14:49:26.116660] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[0.06     , 0.       , 9.7540184, 0.2      ]]), 'metadata': <time_series_
↪metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.116479] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[0.06     , 0.       , 9.7540184, 0.2      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3cee50>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:26.115684] (Sensor4): Sending: [array([[0.06     , 0.       , 9.
↪7540184, 0.2      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:26.118065] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]])}
[2021-07-07 14:49:26.119262] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>]}, 'Sensor2': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>]}, 'Sensor3': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>]}, 'Sensor4': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>]}}
[2021-07-07 14:49:26.119350] (MonitorAgent_SensorValues): Tproc: 0.002783
[2021-07-07 14:49:26.118130] (RedundancyAgent1): Tproc: 0.001394
Buffer size is  7 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:27.007522] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.07      , 0.        , 9.64395692, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3ceeb0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.003888] (Sensor1): Pack time: 0.000931
[2021-07-07 14:49:27.007780] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.07      , 0.        , 9.64395692, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.006537] (Sensor1): Sending: [array([[0.07      , 0.        , 9.
↪64395692, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:27.064438] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>]}, 'Sensor3': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>]}, 'Sensor4': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>]}}
[2021-07-07 14:49:27.049495] (Sensor2): Pack time: 0.00217
[2021-07-07 14:49:27.082978] (Sensor3): Pack time: 0.000381
[2021-07-07 14:49:27.093177] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]])}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:27.065373] (MonitorAgent_SensorValues): Tproc: 0.051613
[2021-07-07 14:49:27.053082] (Sensor2): Sending: [array([[ 0.07      ,  0.        ,␣
↪10.10629053,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:27.086959] (Sensor3): Sending: [array([[ 0.07      ,  0.        ,␣
↪10.39382334,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
[2021-07-07 14:49:27.096507] (RedundancyAgent1): Tproc: 0.088083
[2021-07-07 14:49:27.085681] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>]}, 'Sensor4': {'data': array([[0.
↪00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>]}}
[2021-07-07 14:49:27.097338] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[ 0.07      ,  0.        , 10.10629053,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.073893] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[ 0.07      ,  0.        , 10.10629053,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3cecd0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.108499] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]])}
[2021-07-07 14:49:27.085880] (MonitorAgent_SensorValues): Tproc: 0.011452
[2021-07-07 14:49:27.108738] (RedundancyAgent1): Tproc: 0.011242
[2021-07-07 14:49:27.087157] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.07      ,  0.        , 10.39382334,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3ceb80>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.116905] (Sensor4): Pack time: 0.000174
[2021-07-07 14:49:27.110424] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[ 0.07      ,  0.        , 10.39382334,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:27.095596] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>]}}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:27.117323] (Sensor4): Sending: [array([[0.07     , 0.       , 9.
↪5119209, 0.2      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:27.113833] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01]])}
[2021-07-07 14:49:27.095824] (MonitorAgent_SensorValues): Tproc: 0.008473
[2021-07-07 14:49:27.114747] (RedundancyAgent1): Tproc: 0.004163
[2021-07-07 14:49:27.118762] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[0.07     , 0.       , 9.5119209, 0.2      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad3cefa0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.118666] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[0.07     , 0.       , 9.5119209, 0.2      ]]), 'metadata': <time_series_
↪metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:27.127116] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad41fe80>, <
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.81226671e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cefa0>]}}
[2021-07-07 14:49:27.122042] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]])}
[2021-07-07 14:49:27.127213] (MonitorAgent_SensorValues): Tproc: 0.00831
[2021-07-07 14:49:27.122187] (RedundancyAgent1): Tproc: 0.003364
Buffer size is  8 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 14:49:27.998756] (Sensor1): Pack time: 0.000271
[2021-07-07 14:49:27.999481] (Sensor1): Sending: [array([[0.08     , 0.       , 9.
↪9701676, 0.2      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad47d550>]
[2021-07-07 14:49:28.000684] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.08     , 0.       , 9.9701676, 0.2      ]]), 'metadata': <time_series_
↪metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.002045] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.08     , 0.       , 9.9701676, 0.2      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad459d00>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.008939] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad459d00>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cefa0>]}}
[2021-07-07 14:49:28.010147] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]])}
[2021-07-07 14:49:28.009079] (MonitorAgent_SensorValues): Tproc: 0.006906
[2021-07-07 14:49:28.011280] (RedundancyAgent1): Tproc: 0.010427
[2021-07-07 14:49:28.041395] (Sensor2): Pack time: 0.000193
[2021-07-07 14:49:28.042955] (MonitorAgent_SensorValues): Received: {'from': 'Sensor2
↪', 'data': array([[ 0.08     ,  0.       , 10.3098588,  0.2      ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3f0130>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.041970] (Sensor2): Sending: [array([[ 0.08     ,  0.       , 10.
↪3098588,  0.2      ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4806d0>]
[2021-07-07 14:49:28.044622] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad459d00>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459a10>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0130>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.
↪00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cefa0>]}}
[2021-07-07 14:49:28.045903] (RedundancyAgent1): Received: {'from': 'Sensor2', 'data':
↪ array([[ 0.08     ,  0.       , 10.3098588,  0.2      ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b1c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.048715] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]])}
[2021-07-07 14:49:28.044681] (MonitorAgent_SensorValues): Tproc: 0.001629
[2021-07-07 14:49:28.048806] (RedundancyAgent1): Tproc: 0.002818
[2021-07-07 14:49:28.076686] (Sensor3): Pack time: 0.000103
[2021-07-07 14:49:28.076910] (Sensor3): Sending: [array([[ 0.08      ,  0.        ,␣
↪10.53250935,  0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad4826d0>]
[2021-07-07 14:49:28.077554] (RedundancyAgent1): Received: {'from': 'Sensor3', 'data':
↪ array([[ 0.08      ,  0.        , 10.53250935,  0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.080638] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.05325093e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]])}
```

(continues on next page)

(continued from previous page)

```
[2021-07-07 14:49:28.080717] (RedundancyAgent1): Tproc: 0.003086
[2021-07-07 14:49:28.077555] (MonitorAgent_SensorValues): Received: {'from': 'Sensor3
↪', 'data': array([[ 0.08       ,  0.        , 10.53250935,  0.2       ]]), 'metadata
↪': <time_series_metadata.scheme.MetaData object at 0x7fecad3f0070>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.080644] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad459d00>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459c70>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0130>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.05325093e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0070>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cefa0>]}}
[2021-07-07 14:49:28.080716] (MonitorAgent_SensorValues): Tproc: 0.00308
[2021-07-07 14:49:28.114989] (Sensor4): Pack time: 0.00015
[2021-07-07 14:49:28.115411] (Sensor4): Sending: [array([[0.08      , 0.        , 9.
↪83409251, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad482700>]
[2021-07-07 14:49:28.116198] (RedundancyAgent1): Received: {'from': 'Sensor4', 'data':
↪ array([[0.08      , 0.        , 9.83409251, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7fecad46b3d0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.116198] (MonitorAgent_SensorValues): Received: {'from': 'Sensor4
↪', 'data': array([[0.08      , 0.        , 9.83409251, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7fecad3f0040>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 14:49:28.119805] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[0.00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce6a0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cea00>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ce8b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cea90>, <time_series_metadata.scheme.MetaData object at 0x7fecad44fe80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459b20>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce4c0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceeb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad459d00>]}, 'Sensor2': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce7f0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceaf0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cebb0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced30>, <time_series_metadata.scheme.MetaData object at 0x7fecad459e70>, <
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459af0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce670>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cecd0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0130>]}, 'Sensor3': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪40049983e+00, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.05325093e+01, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3cea30>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce700>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec10>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3ced90>, <time_series_metadata.scheme.MetaData object at 0x7fecad459b80>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459970>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ce8e0>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3ceb80>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0070>]}, 'Sensor4': {'data': array([[0.00000000e+00, 0.00000000e+00, 9.
↪63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.83409251e+00, 2.00000000e-01]]), 'metadata':
↪ [<time_series_metadata.scheme.MetaData object at 0x7fecad3ce730>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3ceb20>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cec70>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3cedf0>, <time_series_metadata.scheme.MetaData object at 0x7fecad459d60>,
↪<time_series_metadata.scheme.MetaData object at 0x7fecad459bb0>, <time_series_
↪metadata.scheme.MetaData object at 0x7fecad3cee50>, <time_series_metadata.scheme.
↪MetaData object at 0x7fecad3cefa0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7fecad3f0040>]}}
[2021-07-07 14:49:28.119751] (RedundancyAgent1): Buffer: {'Sensor1': array([[0.
↪00000000e+00, 0.00000000e+00, 1.00575868e+01, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.85659121e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.99946200e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00805776e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.66887351e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 9.71880656e+00, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.70455528e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.64395692e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.97016760e+00, 2.00000000e-01]]), 'Sensor2':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.81226671e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.62458464e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.85543299e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.00927391e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 9.87107856e+00, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00922106e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.01766134e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.01062905e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.03098588e+01, 2.00000000e-01]]), 'Sensor3':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.40049983e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.33105188e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 9.72930289e+00, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.01384766e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.00575535e+01, 2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [5.00000000e-02, 0.00000000e+00, 1.03679675e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 1.04846347e+01, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 1.03938233e+01, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 1.05325093e+01, 2.00000000e-01]]), 'Sensor4':␣
↪array([[0.00000000e+00, 0.00000000e+00, 9.63403863e+00, 2.00000000e-01],
       [1.00000000e-02, 0.00000000e+00, 9.81544544e+00, 2.00000000e-01],
       [2.00000000e-02, 0.00000000e+00, 1.03398175e+01, 2.00000000e-01],
       [3.00000000e-02, 0.00000000e+00, 1.06398883e+01, 2.00000000e-01],
       [4.00000000e-02, 0.00000000e+00, 1.02079757e+01, 2.00000000e-01],
       [5.00000000e-02, 0.00000000e+00, 1.00425699e+01, 2.00000000e-01],
       [6.00000000e-02, 0.00000000e+00, 9.75401840e+00, 2.00000000e-01],
       [7.00000000e-02, 0.00000000e+00, 9.51192090e+00, 2.00000000e-01],
       [8.00000000e-02, 0.00000000e+00, 9.83409251e+00, 2.00000000e-01]])}
[2021-07-07 14:49:28.120010] (MonitorAgent_SensorValues): Tproc: 0.003683
[2021-07-07 14:49:28.119848] (RedundancyAgent1): Tproc: 0.003522
Buffer size is  9 , which is less than  10 .
Not enough data for redundancy agent evaluation.
Buffer size is  9 , which is less than  10 .
Not enough data for redundancy agent evaluation.
```

7.2 Redundancy Agent – Determining redundancy in one single signal

In this tutorial we generate a single signal containing redundant, correlated information and supply it to the Redundancy Agent. The agent uses the redundancy in the data vector together with some prior knowledge in order to calculate the best consistent estimate with associated uncertainty, respecting all provided input uncertainties. It rejects sensor values that may be erroneous. In this presented use case the sensors do not directly measure the measurand, but the measurand is linked to the sensor values by means of four linear equations. The fact that there are four equations and not just one is the cause of the redundancy.

At the start of the main module we define all important parameters. For detailed descriptions of the parameters please refer to the docstrings of the respective methods. Then we define the agents and start the network. The network and the calculated results can be monitored in a browser at the address http://127.0.0.1:8050/.

[2]:
```
# %load redundancy_agent_one_signal.py
import numpy as np

from agentMET4FOF.agents.metrological_redundancy_agents import RedundancyAgent
from agentMET4FOF.agents.metrological_base_agents import (
    MetrologicalMonitorAgent,
)
from agentMET4FOF.agents.metrological_signal_agents import (
    MetrologicalGeneratorAgent,
)
from agentMET4FOF.network import AgentNetwork
from agentMET4FOF.streams.metrological_signal_streams import (
    MetrologicalMultiWaveGenerator,
)


def demonstrate_redundancy_agent_onesignal():
    batch_size = 10
    n_pr = batch_size
```

(continues on next page)

(continued from previous page)

```
    sampling_frequency = 40
    signal_1_frequency = 6
    signal_2_frequency = 10
    signal_1_initial_phase = 1
    signal_2_initial_phase = 2
    signal_1_amplitude = 230
    signal_2_amplitude = 20
    exp_unc_abs = 0.2  # absolute expanded uncertainty
    probability_limit = 0.95

    # start agent network server
    agent_network = AgentNetwork(dashboard_modules=True)

    # Initialize signal generating class outside of agent framework.
    signal1 = MetrologicalMultiWaveGenerator(
        sfreq=sampling_frequency,
        freq_arr=np.array([signal_1_frequency, signal_2_frequency]),
        ampl_arr=np.array([signal_1_amplitude, signal_2_amplitude]),
        phase_ini_arr=np.array([signal_1_initial_phase, signal_2_initial_phase]),
        value_unc=exp_unc_abs,
    )
    # signal1.init_parameters(batch_size1=batch_size)

    # Data source agents.
    source_name1 = "Sensor1"  # signal1.metadata.metadata["device_id"]
    source_agent1 = agent_network.add_agent(
        name=source_name1, agentType=MetrologicalGeneratorAgent
    )
    source_agent1.init_parameters(signal=signal1, batch_size=batch_size)

    # Redundant data processing agent
    sensor_key_list = [source_name1]
    redundancy_name1 = "RedundancyAgent1"  # Name cannot contain spaces!!
    redundancy_agent1 = agent_network.add_agent(
        name=redundancy_name1, agentType=RedundancyAgent
    )
    redundancy_agent1.init_parameters(
        sensor_key_list=sensor_key_list,
        calc_type="lcss",
        n_pr=n_pr,
        problim=probability_limit,
    )
    # prior knowledge needed for redundant evaluation of the data
    redundancy_agent1.init_lcss_parameters(
        fsam=sampling_frequency,
        f1=signal_1_frequency,
        f2=signal_2_frequency,
        ampl_ratio=signal_1_amplitude / signal_2_amplitude,
        phi1=signal_1_initial_phase,
        phi2=signal_2_initial_phase,
    )

    # Initialize metrologically enabled plotting agent.(
    monitor_agent1 = agent_network.add_agent(
        name="MonitorAgent_SensorValues", agentType=MetrologicalMonitorAgent
    )
    monitor_agent2 = agent_network.add_agent(
```

(continues on next page)

(continued from previous page)

```
        name="MonitorAgent_RedundantEstimate", agentType=MetrologicalMonitorAgent
    )

    # Bind agents.
    source_agent1.bind_output(monitor_agent1)
    source_agent1.bind_output(redundancy_agent1)
    redundancy_agent1.bind_output(monitor_agent2)

    # Set all agents states to "Running".
    agent_network.set_running_state()

    # Allow for shutting down the network after execution.
    return agent_network


if __name__ == "__main__":
    demonstrate_redundancy_agent_onesignal()
```

```
Starting NameServer...
Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:3333 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:3333

------------------------------------------------------------
|                                                          |
| Your agent network is starting up. Open your browser and |
| visit the agentMET4FOF dashboard on http://127.0.0.1:8050/ |
|                                                          |
------------------------------------------------------------

INFO [2021-07-07 15:01:10.413429] (Sensor1): INITIALIZED
INFO [2021-07-07 15:01:10.450809] (RedundancyAgent1): INITIALIZED
INFO [2021-07-07 15:01:10.484041] (MonitorAgent_SensorValues): INITIALIZED
INFO [2021-07-07 15:01:10.524346] (MonitorAgent_RedundantEstimate): INITIALIZED
[2021-07-07 15:01:10.546007] (Sensor1): Connected output module: MonitorAgent_
↪SensorValues
[2021-07-07 15:01:10.559178] (Sensor1): Connected output module: RedundancyAgent1
[2021-07-07 15:01:10.574231] (RedundancyAgent1): Connected output module:␣
↪MonitorAgent_RedundantEstimate
/home/ludwig10/code/agentMET4FOF/agentMET4FOF/metrological_streams.py:135:␣
↪UserWarning:

No uncertainty generator function specified. Setting to default (value_unc = constant,
↪ time_unc = 0).

SET STATE:   Running
[2021-07-07 15:01:11.424478] (Sensor1): Pack time: 0.001013
[2021-07-07 15:01:11.428880] (Sensor1): Sending: [array([[  0.        ,   0.        ,␣
↪115.89771535,   0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7be50>]
[2021-07-07 15:01:11.440180] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[  0.        ,   0.        , 115.89771535,   0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7f4d32e917c0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 15:01:11.451151] (RedundancyAgent1): Buffer: {'Sensor1': array([[  0.    ␣
↪    ,   0.        , 115.89771535,   0.2       ]])}
```

(continues on next page)

(continued from previous page)

```
Buffer size is [2021-07-07 15:01:11.453337] (MonitorAgent_SensorValues): Received: {
↪'from': 'Sensor1', 'data': array([[  0.        ,   0.        , 115.89771535,   0.2
↪     ]]), 'metadata': <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e933d0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
 [2021-07-07 15:01:11.451664] (RedundancyAgent1): Tproc: 0.003382
1 [2021-07-07 15:01:11.456280] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data
↪': array([[  0.        ,   0.        , 115.89771535,   0.2       ]]), 'metadata': [
↪<time_series_metadata.scheme.MetaData object at 0x7f4d32e933d0>]}}
, which is less than  [2021-07-07 15:01:11.456832] (MonitorAgent_SensorValues): Tproc:
↪ 0.002794
10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 15:01:12.419936] (Sensor1): Pack time: 0.000606
[2021-07-07 15:01:12.422322] (Sensor1): Sending: [array([[ 2.50000000e-02,  0.
↪00000000e+00, -1.01535705e+02,
         2.00000000e-01]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7be50>]
[2021-07-07 15:01:12.423682] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e917c0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
[2021-07-07 15:01:12.423671] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e93490>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
Buffer size is [2021-07-07 15:01:12.428294] (MonitorAgent_SensorValues): Buffer: {
↪'Sensor1': {'data': array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
        [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>
↪]}}
 [2021-07-07 15:01:12.426791] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
        [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01]])}
2[2021-07-07 15:01:12.428951] (MonitorAgent_SensorValues): Tproc: 0.003767
 , which is less than  [2021-07-07 15:01:12.427470] (RedundancyAgent1): Tproc: 0.
↪003275
10 .
Not enough data for redundancy agent evaluation.
NS shut down.
[2021-07-07 15:01:13.420547] (Sensor1): Pack time: 0.000617
[2021-07-07 15:01:13.422979] (Sensor1): Sending: [array([[ 5.00000000e-02,  0.
↪00000000e+00, -2.14017298e+02,
         2.00000000e-01]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7be50>]
[2021-07-07 15:01:13.423901] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e722b0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
Buffer size is [2021-07-07 15:01:13.424103] (MonitorAgent_SensorValues): Received: {
↪'from': 'Sensor1', 'data': array([[ 5.00000000e-02,  0.00000000e+00, -2.
↪14017298e+02,
```

(continues on next page)

(continued from previous page)

```
        2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e72070>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
 [2021-07-07 15:01:13.427524] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data
↪': array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
        2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
        2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
        2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>,
↪ <time_series_metadata.scheme.MetaData object at 0x7f4d32e72070>]}}
3[2021-07-07 15:01:13.435063] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪00000000e+00,  0.00000000e+00,  1.15897715e+02,
        2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
        2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
        2.00000000e-01]])}
 , which is less than [2021-07-07 15:01:13.427924] (MonitorAgent_SensorValues): Tproc:
↪ 0.003115
 10. [2021-07-07 15:01:13.435851] (RedundancyAgent1): Tproc: 0.010717

Not enough data for redundancy agent evaluation.
[2021-07-07 15:01:14.419976] (Sensor1): Pack time: 0.000621
[2021-07-07 15:01:14.422191] (Sensor1): Sending: [array([[ 7.50000000e-02,  0.
↪00000000e+00, -1.59767167e+02,
        2.00000000e-01]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7be50>]
[2021-07-07 15:01:14.423266] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
        2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e722b0>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
[2021-07-07 15:01:14.423510] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
        2.00000000e-01]]), 'metadata': <time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e72580>, 'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default
↪'}
[2021-07-07 15:01:14.431719] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
        2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
        2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
        2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
        2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>,
↪ <time_series_metadata.scheme.MetaData object at 0x7f4d32e72070>, <time_series_
↪metadata.scheme.MetaData object at 0x7f4d32e72580>]}}
Buffer size is [2021-07-07 15:01:14.427207] (RedundancyAgent1): Buffer: {'Sensor1':␣
↪array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
        2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
        2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
```

(continues on next page)

(continued from previous page)

```
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01]])}
 [2021-07-07 15:01:14.432797] (MonitorAgent_SensorValues): Tproc: 0.008584
4 [2021-07-07 15:01:14.428017] (RedundancyAgent1): Tproc: 0.003956
, which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 15:01:15.419689] (Sensor1): Pack time: 0.000506
[2021-07-07 15:01:15.421859] (Sensor1): Sending: [array([[0.1       , 0.        , 4.
↪97982699, 0.2       ]]), <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7be50>]
[2021-07-07 15:01:15.423037] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[0.1       , 0.        , 4.97982699, 0.2       ]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7f4d32e7a5b0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 15:01:15.423106] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[0.1       , 0.        , 4.97982699, 0.2       ]]), 'metadata': <time_
↪series_metadata.scheme.MetaData object at 0x7f4d32e722b0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
Buffer size is  [2021-07-07 15:01:15.428496] (RedundancyAgent1): Buffer: {'Sensor1':␣
↪array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01]])}
5[2021-07-07 15:01:15.429141] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data
↪': array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>,
↪ <time_series_metadata.scheme.MetaData object at 0x7f4d32e72070>, <time_series_
↪metadata.scheme.MetaData object at 0x7f4d32e72580>, <time_series_metadata.scheme.
↪MetaData object at 0x7f4d32e7a5b0>]}}
  , which is less than [2021-07-07 15:01:15.429139] (RedundancyAgent1): Tproc: 0.
↪005354
10 .
[2021-07-07 15:01:15.430533] (MonitorAgent_SensorValues): Tproc: 0.006765
Not enough data for redundancy agent evaluation.
[2021-07-07 15:01:16.421513] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[1.2500000e-01, 0.0000000e+00, 1.7541884e+02, 2.0000000e-01]]),
↪'metadata': <time_series_metadata.scheme.MetaData object at 0x7f4d32e7a5e0>,
↪'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 15:01:16.423369] (MonitorAgent_SensorValues): Buffer: {'Sensor1': {'data':
↪ array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01],
       [ 1.25000000e-01,  0.00000000e+00,  1.75418840e+02,
         2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>,
↪ <time_series_metadata.scheme.MetaData object at 0x7f4d32e72070>, <time_series_
↪metadata.scheme.MetaData object at 0x7f4d32e72580>, <time_series_metadata.scheme.
↪MetaData object at 0x7f4d32e7a5b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7a5e0>]}}
[2021-07-07 15:01:16.422533] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[1.2500000e-01, 0.0000000e+00, 1.7541884e+02, 2.0000000e-01]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7f4d32e722b0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 15:01:16.419512] (Sensor1): Pack time: 0.000535
[2021-07-07 15:01:16.423744] (MonitorAgent_SensorValues): Tproc: 0.002008
[2021-07-07 15:01:16.428532] (RedundancyAgent1): Buffer: {'Sensor1': array([[ 0.
↪00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01],
       [ 1.25000000e-01,  0.00000000e+00,  1.75418840e+02,
         2.00000000e-01]])}
[2021-07-07 15:01:16.425628] (Sensor1): Sending: [array([[1.2500000e-01, 0.
↪0000000e+00, 1.7541884e+02, 2.0000000e-01]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7f4d32e7be50>]
[2021-07-07 15:01:16.428746] (RedundancyAgent1): Tproc: 0.00601
Buffer size is  6 , which is less than  10 .
Not enough data for redundancy agent evaluation.
[2021-07-07 15:01:17.440260] (RedundancyAgent1): Received: {'from': 'Sensor1', 'data':
↪ array([[1.5000000e-01, 0.0000000e+00, 2.2239348e+02, 2.0000000e-01]]), 'metadata':
↪<time_series_metadata.scheme.MetaData object at 0x7f4d32e722b0>, 'senderType':
↪'MetrologicalGeneratorAgent', 'channel': 'default'}
Buffer size is [2021-07-07 15:01:17.450576] (RedundancyAgent1): Buffer: {'Sensor1':␣
↪array([[ 0.00000000e+00,  0.00000000e+00,  1.15897715e+02,
         2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01],
       [ 1.25000000e-01,  0.00000000e+00,  1.75418840e+02,
         2.00000000e-01],
```

(continues on next page)

(continued from previous page)

```
       [ 1.50000000e-01,  0.00000000e+00,  2.22393480e+02,
         2.00000000e-01]])}
[2021-07-07 15:01:17.446961] (Sensor1): Pack time: 0.02075
 [2021-07-07 15:01:17.453076] (MonitorAgent_SensorValues): Received: {'from': 'Sensor1
↪', 'data': array([[1.5000000e-01, 0.0000000e+00, 2.2239348e+02, 2.0000000e-01]]),
↪'metadata': <time_series_metadata.scheme.MetaData object at 0x7f4d32e7a880>,
↪'senderType': 'MetrologicalGeneratorAgent', 'channel': 'default'}
[2021-07-07 15:01:17.463997] (RedundancyAgent1): Tproc: 0.016613
7 [2021-07-07 15:01:17.463909] (Sensor1): Sending: [array([[1.5000000e-01, 0.
↪0000000e+00, 2.2239348e+02, 2.0000000e-01]]), <time_series_metadata.scheme.MetaData␣
↪object at 0x7f4d32e7be50>]
, which is less than  [2021-07-07 15:01:17.463269] (MonitorAgent_SensorValues):␣
↪Buffer: {'Sensor1': {'data': array([[ 0.00000000e+00,  0.00000000e+00,  1.
↪15897715e+02,
         2.00000000e-01],
       [ 2.50000000e-02,  0.00000000e+00, -1.01535705e+02,
         2.00000000e-01],
       [ 5.00000000e-02,  0.00000000e+00, -2.14017298e+02,
         2.00000000e-01],
       [ 7.50000000e-02,  0.00000000e+00, -1.59767167e+02,
         2.00000000e-01],
       [ 1.00000000e-01,  0.00000000e+00,  4.97982699e+00,
         2.00000000e-01],
       [ 1.25000000e-01,  0.00000000e+00,  1.75418840e+02,
         2.00000000e-01],
       [ 1.50000000e-01,  0.00000000e+00,  2.22393480e+02,
         2.00000000e-01]]), 'metadata': [<time_series_metadata.scheme.MetaData object␣
↪at 0x7f4d32e933d0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e93490>,
↪ <time_series_metadata.scheme.MetaData object at 0x7f4d32e72070>, <time_series_
↪metadata.scheme.MetaData object at 0x7f4d32e72580>, <time_series_metadata.scheme.
↪MetaData object at 0x7f4d32e7a5b0>, <time_series_metadata.scheme.MetaData object at␣
↪0x7f4d32e7a5e0>, <time_series_metadata.scheme.MetaData object at 0x7f4d32e7a880>]}}
10 [2021-07-07 15:01:17.463567] (MonitorAgent_SensorValues): Tproc: 0.010296
.
Not enough data for redundancy agent evaluation.
```

CHAPTER 8

agentMET4FOF agents

8.1 Base agents

class agentMET4FOF.agents.base_agents.**AgentMET4FOF**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

Base class for all agents with specific functions to be overridden/supplied by user.

Behavioral functions for users to provide are init_parameters, agent_loop and on_received_message. Communicative functions are bind_output, unbind_output and send_output.

_bind_output(*output_agent*, *channel='default'*)

Internal method which implements the logic for connecting this agent, to the *output_agent*.

_convert_matplotlib_fig(*fig: matplotlib.figure.Figure*, *mode: str = 'image'*)

Convert matplotlib figure to be rendered by the dashboard

_convert_to_plotly(*matplotlib_fig: matplotlib.figure.Figure*)

Internal method to convert matplotlib figure to plotly figure

Parameters **matplotlib_fig** (*plt.Figure*) – Matplotlib figure to be converted

_fig_to_uri(*matplotlib_fig: matplotlib.figure.Figure*)

Internal method to convert matplotlib figure to base64 uri image for display

Parameters **matplotlib_fig** (*plt.Figure*) – Matplotlib figure to be converted

_get_metadata(*data*)

Internal helper function for getting the data type & dimensions of data. This is for update_output_channels_info()

_is_type_message(*data*)

Internal method to check if the data carries signature of an agent message type

Parameters **data** – Data to be checked for type

Returns result

Return type boolean

_remove_methods(*cls*)
Remove methods from the other backends base class from the current agent

_update_output_channels_info(*data*, *channel*)
Internal method to update the dict of output_channels_info. This is used in conjunction with send_output().

Checks and records data type & dimension and channel name If the data is nested within dict, then it will search deeper and subsequently record the info of each inner hierarchy

Parameters

- **data** – data to be checked for type & dimension
- **channel** (*str*) – name of channel to be recorded

agent_loop()
User defined method for the agent to execute for *loop_wait* seconds specified either in *self.loop_wait* or explicitly via *init_agent_loop(loop_wait)*

To start a new loop, call *init_agent_loop(loop_wait)* on the agent. Example of usage is to check the *current_state* of the agent and send data periodically.

bind_output(*output_agent*, *channel='default'*)
Forms Output connection with another agent

Any call on send_output will reach this newly binded agent. Adds the agent to its list of Outputs.

Parameters

- **output_agent** (*AgentMET4FOF or list*) – Agent(s) to be binded to this agent's output channel
- **channel** (*str or list of str*) – Specific name of the channel(s) to be subscribed to. (Default = "data")

buffer_clear(*agent_name: Optional[str] = None*)
Empties buffer which is a dict indexed by the *agent_name*.

Parameters **agent_name** (*str, optional*) – Key of the memory dict, which can be the name of input agent, or self.name. If not supplied (default), we assume to clear the entire memory.

buffer_filled(*agent_name=None*)
Checks whether the internal buffer has been filled to the maximum allowed specified by self.buffer_size

Parameters **agent_name** (*str*) – Index of the buffer which is the name of input agent.

Returns status of buffer filled

Return type boolean

buffer_store(*agent_from: str*, *data=None*, *concat_axis=0*)
Updates data stored in *self.buffer* with the received message

Checks if sender agent has sent any message before If it did,then append, otherwise create new entry for it

Parameters

- **agent_from** (*str*) – Name of agent sender
- **data** – Any supported data which can be stored in dict as buffer. See AgentBuffer for more information.

get_attr(*attr*)
Return the specified attribute of the agent.

Parameters **name** – Name of the attribute to be retrieved.

handle_process_data(*message*)
Internal method to handle incoming message before calling on_received_message

If current_state is either Stop or Reset, it will terminate early before entering on_received_message.

init_agent(*buffer_size=1000*, *log_mode=True*)
Internal initialization to setup the agent: mainly on setting the dictionary of Inputs, Outputs, PubAddr. Calls user-defined *init_parameters()* upon finishing.

Inputs
Dictionary of Agents connected to its input channels. Messages will arrive from agents in this dictionary. Automatically updated when *bind_output()* function is called
Type dict

Outputs
Dictionary of Agents connected to its output channels. Messages will be sent to agents in this dictionary. Automatically updated when *bind_output()* function is called
Type dict

PubAddr_alias
Name of Publish address socket
Type str

PubAddr
Publish address socket handle
Type str

AgentType
Name of class
Type str

current_state
Current state of agent. Can be used to define different states of operation such as "Running", "Idle, "Stop", etc.. Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*
Type str

loop_wait
The interval to wait between loop. Call *init_agent_loop* to restart the timer or set the value of loop_wait in *init_parameters* when necessary.
Type int

buffer_size
The total number of elements to be stored in the agent `buffer` When total elements exceeds this number, the latest elements will be replaced with the incoming data elements
Type int

init_agent_loop(*loop_wait: Optional[int] = None*)
Initiates the agent loop, which iterates every *loop_wait* seconds

Stops every timers and initiate a new loop.

Parameters **loop_wait** (*int, optional*) – The wait between each iteration of the loop

init_buffer(*buffer_size*)
A method to initialise the buffer. By overriding this method, user can provide a custom buffer, instead

of the regular AgentBuffer. This can be used, for example, to provide a MetrologicalAgentBuffer in the metrological agents.

init_parameters()
User provided function to initialize parameters of choice.

log_info(*message*)
Prints logs to be saved into logfile with Logger Agent

Parameters **message** (*str*) – Message to be logged to the internal Logger Agent

on_connect_output(*output_agent*)
This method is called whenever an agent is connected to its output

This can be for example, to send *metadata* or *ping* to the output agent.

on_received_message(*message*)
User-defined method and is triggered to handle the message passed by Input.

Parameters **message** (*Dictionary*) – The message received is in form {'from':agent_name, 'data': data, 'senderType': agent_class, 'channel':channel_name}. agent_name is the name of the Input agent which sent the message data is the actual content of the message.

pack_data(*data*, *channel='default'*)
Internal method to pack the data content into a dictionary before sending out.

Special case : if the *data* is already a *message*, then the *from* and *senderType* will be altered to this agent, without altering the *data* and *channel* within the message this is used for more succinct data processing and passing.

Parameters

- **data** (*argument*) – Data content to be packed before sending out to agents.
- **channel** (*str*) – Key of dictionary which stores data

Returns

- **Packed message data** (*dict of the form {'from':agent_name, 'data': data,*)
- **'senderType'** (*agent_class, 'channel':channel_name}.*)

reset()
Reset the agent's states and parameters

User can override this method to reset the specific parameters.

respond_reply_attr_(*message_data*)
Response to a *reply* of setting attribute

respond_request_attr_(*attribute: str*)
Response to a *request* of *attribute* from input agents.

This agent reply with the requested *attribute* if it has it.

respond_request_method_(*message_data: dict*)
Response to a *request* of executing *method* from input agents.

This agent will execute the method with the provided parameters of the method.

send_output(*data*, *channel='default'*)
Sends message data to all connected agents in self.Outputs.

Output connection can first be formed by calling bind_output. By default calls pack_data(data) before sending out. Can specify specific channel as opposed to 'default' channel.

Parameters

- **data** (*argument*) – Data content to be sent out
- **channel** (*str*) – Key of *message* dictionary which stores data

Returns message – {'from':agent_name, 'data': data, 'senderType': agent_class, 'channel':channel_name}.

Return type dict

send_plot(*fig: Union[matplotlib.figure.Figure, Dict[str, matplotlib.figure.Figure]], mode: str = 'image'*)

Sends plot to agents connected to this agent's Output channel.

This method is different from send_output which will be sent to through the 'plot' channel to be handled.

Tradeoffs between "image" and "plotly" modes are that "image" are more stable and "plotly" are interactive. Note not all (complicated) matplotlib figures can be converted into a plotly figure.

Parameters

- **fig** (*matplotlib.figure.Figure or dict of matplotlib.figure.Figure*) – Alternatively, multiple figures can be nested in a dict (with any preferred keys) e.g {"Temperature":matplotlib.Figure, "Acceleration":matplotlib.Figure}
- **mode** (*str*) – "image" - converts into image via encoding at base64 string. "plotly" - converts into plotly figure using *mpl_to_plotly* Default: "image"

Returns graph

Return type str or plotly figure or dict of one of those converted figure(s)

send_request_attribute(*attribute: str*)

Send a *request* of *attribute* to output agents.

Output agents will reply with the requested *attribute* if they have.

send_request_method(*method: str*, ***method_params*)

Send a *request* of executing methods to output agents.

Output agents will respond by calling the method.

send_set_attr(*attr: str*, *value*)

Sends a message to set the *attr* of another agent to that of *value*.

Parameters

- **attr** (*str*) – The variable name of the output agent to be set.
- **value** – The value of the variable to be set

set_attr(***kwargs*)

Set object attributes.

Parameters **kwargs** (*[name, value]*) – Keyword arguments will be used to set the object attributes.

shutdown()

Cleanly stop and shut down the agent assuming the agent is running.

Will let the main thread do the tear down.

step()

Used for MESA backend only. Behaviour on every update step.

stop_agent_loop()
Stops agent_loop from running

Note that the agent will still be responding to messages.

unbind_output(*output_agent*)
Remove existing output connection with another agent

This reverses the bind_output method.

Parameters **output_agent** (AgentMET4FOF) – Agent binded to this agent's output channel

class agentMET4FOF.agents.base_agents.**DataStreamAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)
Able to simulate generation of datastream by loading a given DataStreamMET4FOF

Can be used in incremental training or batch training mode. To simulate batch training mode, set *pretrain_size=-1* , otherwise, set pretrain_size and batch_size for the respective. See *DataStreamMET4FOF* on loading your own data set as a data stream.

agent_loop()
User defined method for the agent to execute for *loop_wait* seconds specified either in *self.loop_wait* or explicitly via *init_agent_loop(loop_wait)*

To start a new loop, call *init_agent_loop(loop_wait)* on the agent. Example of usage is to check the *current_state* of the agent and send data periodically.

init_parameters(*stream=<agentMET4FOF.streams.base_streams.DataStreamMET4FOF object>*, *pretrain_size=None*, *batch_size=1*, *loop_wait=1*, *randomize=False*)

Parameters

- **stream** (DataStreamMET4FOF) – A DataStreamMET4FOF object which provides the sample data
- **pretrain_size** (*int*) – The number of sample data to send through in the first loop cycle, and subsequently, the batch_size will be used
- **batch_size** (*int*) – The number of sample data to send in every loop cycle
- **loop_wait** (*int*) – The duration to wait (seconds) at the end of each loop cycle before going into the next cycle
- **randomize** (*bool*) – Determines if the dataset should be shuffled before streaming

reset()
Reset the agent's states and parameters

User can override this method to reset the specific parameters.

class agentMET4FOF.agents.base_agents.**MonitorAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)
Unique Agent for storing plots and data from messages received from input agents.

The dashboard searches for Monitor Agents' *buffer* and *plots* to draw the graphs "plot" channel is used to receive base64 images from agents to plot on dashboard

plots
Dictionary of format *{agent1_name : agent1_plot, agent2_name : agent2_plot}*

Type dict

plot_filter
List of keys to filter the 'data' upon receiving message to be saved into memory Used to specifically select only a few keys to be plotted

Type list of str

custom_plot_function
a custom plot function that can be provided to handle the data in the monitor agents buffer (see *AgentMET4FOF* for details). The function gets provided with the content (value) of the buffer and with the string of the sender agent's name as stored in the buffer's keys. Additionally any other parameters can be provided as a dict in custom_plot_parameters.

Type callable

custom_plot_parameters
a custom dictionary of parameters that shall be provided to each call of the custom_plot_function

Type dict

init_parameters(*plot_filter: Optional[List[str]] = None, custom_plot_function: Optional[Callable[[...], plotly.graph_objs._scatter.Scatter]] = None, **kwargs*)
Initialize the monitor agent's parameters

Parameters

- **plot_filter** (*list of str, optional*) – List of keys to filter the 'data' upon receiving message to be saved into memory. Used to specifically select only a few keys to be plotted
- **custom_plot_function** (*callable, optional*) – a custom plot function that can be provided to handle the data in the monitor agents buffer (see *AgentMET4FOF* for details). The function gets provided with the content (value) of the buffer and with the string of the sender agent's name as stored in the buffer's keys. Additionally any other parameters can be provided as a dict in custom_plot_parameters. By default the data gets plotted as shown in the various tutorials.
- **kwargs** (*Any*) – custom key word parameters that shall be provided to each call of the *custom_plot_function*

on_received_message(*message*)
Handles incoming data from 'default' and 'plot' channels.

Stores 'default' data into `buffer` and 'plot' data into *plots*

Parameters **message** (*dict*) – Acceptable channel values are 'default' or 'plot'

reset()
Reset the agent's states and parameters

User can override this method to reset the specific parameters.

update_plot_memory(*message: Dict[str, Any]*)
Updates plot figures stored in *self.plots* with the received message

Parameters **message** (*dict*) – Standard message format specified by AgentMET4FOF class Message['data'] needs to be base64 image string and can be nested in dictionary for multiple plots. Only the latest plot will be shown kept and does not keep a history of the plots.

8.2 Signal agents

class agentMET4FOF.agents.signal_agents.**SineGeneratorAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

An agent streaming a sine signal

Takes samples from the SineGenerator and pushes them sample by sample to connected agents via its output channel.

agent_loop()

Model the agent's behaviour

On state *Running* the agent will extract sample by sample the input data streams content and push it via invoking AgentMET4FOF.send_output().

init_parameters(*sfreq=500*, *sine_freq=5*, *amplitude=1*, *initial_phase=0*)

Initialize the input data

Initialize the input data stream as an instance of the SineGenerator class.

Parameters

- **sfreq** (*int*) – sampling frequency for the underlying signal
- **sine_freq** (*float*) – frequency of the generated sine wave
- **amplitude** (*float*) – amplitude of the generated sine wave
- **initial_phase** (*float*) – initial phase (at t=0) of the generated sine wave

class agentMET4FOF.agents.signal_agents.**StaticSineWithJitterGeneratorAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

An agent streaming a pre generated sine signal of fixed length with jitter

Takes samples from the StaticSineGeneratorWithJitter and pushes them sample by sample to connected agents via its output channel.

agent_loop()

Extract sample by sample the input data stream's content and push it

init_parameters(*num_cycles: Optional[int] = 1000*, *jitter_std: Optional[float] = 0.02*)

Initialize the pre generated sine signal of fixed length with jitter

Initialize the static input data as an instance of the StaticSineWithJitterGenerator class with the provided parameters.

Parameters

- **num_cycles** (*int, optional*) – numbers of cycles, determines the signal length by $\pi \cdot num_cycles$, defaults to 1000
- **jitter_std** (*float, optional*) – the standard deviation of the distribution to randomly draw jitter from, defaults to 0.02

class agentMET4FOF.agents.signal_agents.**NoiseAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

An agent adding white noise to the incoming signal

init_parameters(*noise_std: Optional[float] = 0.05*)

Initialize the noise's standard deviation

Parameters **noise_std** (*float, optional*) – the standard deviation of the distribution to randomly draw noise from, defaults to 0.05

noise_std

Standard deviation of the distribution to randomly draw noise from

on_received_message(*message: Dict[str, Any]*)

Add noise to the received message's data

Parameters **message** (*Dictionary*) – the received message in the expected form:

```
dict like {
    "from": "<valid agent name>"
    "data": <time series data as a list, np.ndarray or pd.Dataframe>
↪ or
        dict like {
            "quantities": <time series data as a list, np.ndarray or
                pd.Dataframe>,
            "target": <target labels as a list, np.ndarray or pd.
↪Dataframe>,
            "time": <time stamps as a list, np.ndarray or pd.
↪Dataframe of
                float or np.datetime64>
        }
    "senderType": <any subclass of AgentMet4FoF>,
    "channel": "<channel name>"
    }
```

8.3 Metrologically enabled base agents

class agentMET4FOF.agents.metrological_base_agents.**MetrologicalAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

_input_data = None
Input dictionary of all incoming data including metadata:

```
dict like {
    <from>: {
        "buffer": TimeSeriesBuffer(maxlen=buffer_size),
        "metadata": MetaData(**kwargs).metadata,
    }
```

_output_data = None
Output dictionary of all outgoing data including metadata:

```
dict like {
    <from>: {
        "buffer": TimeSeriesBuffer(maxlen=buffer_size),
        "metadata": MetaData(**kwargs).metadata,
    }
```

agent_loop()
User defined method for the agent to execute for *loop_wait* seconds specified either in *self.loop_wait* or explicitly via *init_agent_loop(loop_wait)*

To start a new loop, call *init_agent_loop(loop_wait)* on the agent. Example of usage is to check the *current_state* of the agent and send data periodically.

init_parameters(*input_data_maxlen=25*, *output_data_maxlen=25*)
User provided function to initialize parameters of choice.

on_received_message(*message*)
User-defined method and is triggered to handle the message passed by Input.

Parameters **message** (*Dictionary*) – The message received is in form {'from':agent_name, 'data': data, 'senderType': agent_class, 'channel':channel_name}. agent_name is the name of the Input agent which sent the message data is the actual content of the message.

pack_data(*data*, *channel='default'*)
Internal method to pack the data content into a dictionary before sending out.

Special case : if the *data* is already a *message*, then the *from* and *senderType* will be altered to this agent, without altering the *data* and *channel* within the message this is used for more succinct data processing and passing.

Parameters

- **data** (*argument*) – Data content to be packed before sending out to agents.
- **channel** (*str*) – Key of dictionary which stores data

Returns

- **Packed message data** (*dict of the form {'from':agent_name, 'data': data,}*)
- **'senderType'** (*agent_class, 'channel':channel_name}.*)

class agentMET4FOF.agents.metrological_base_agents.**MetrologicalMonitorAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

init_parameters(**args*, ***kwargs*)
User provided function to initialize parameters of choice.

on_received_message(*message*)
Handles incoming data from `default` and `plot` channels

Feeds `default` data into the buffer as a dictionary:

dict like { "data": message["data"], "metadata": message["metadata"],

}

and hands over 'plot' data to plot memory.

Parameters **message** (*dict*) – Acceptable `channel` values are `default` or `plot`

reset()
Reset the agent's states and parameters

User can override this method to reset the specific parameters.

update_plot_memory(*message*)
Updates plot figures stored in *self.plots* with the received message

Parameters **message** (*dict*) – Standard message format specified by AgentMET4FOF class Message['data'] needs to be base64 image string and can be nested in dictionary for multiple plots Only the latest plot will be shown kept and does not keep a history of the plots.

8.4 Metrological agents to reduce redundancy

class agentMET4FOF.agents.metrological_redundancy_agents.**RedundancyAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)

This is the main Redundancy Agent class

Redundancy means that there is more than one way to derive the value of the measurand Y from the values of the sensor data X_i. Following main cases are considered in the agent:

- Redundant measurement of the measurand Y by independent sensors directly measuring Y
- Redundant measurement of the measurand Y by correlated sensors directly measuring Y
- Redundant measurement of the measurand Y by correlated sensors X_i indirectly measuring Y, with a linear relationship y = a + A * x between the vector x of sensor values and the vector y containing the various (redundant) estimates of the measurand Y, where a is a vector and A a matrix both of appropriate size.

Main calculations are performed in `calc_lcs()` and `calc_lcss()`. Usage of the `RedundancyAgent` is relatively straightforward. Note that all static functions have their own test functions illustrating their usage. Details of the different methods are presented in their respective docstrings.

Please refer to other sections in this documentation for more information. A scientific publication explaining the ideas behind this agent can be found in [Kok2020_1]. Related work can be found in [Kok2020_2].

The usage of the Redundancy Agent is illustrated with two examples contained in *two tutorials*.

References

- Kok and Harris [Kok2020_1]
- Kok and Harris [Kok2020_2]

agent_loop()
Model the agent's behaviour

On state *Running* the agent will extract sample by sample the input data streams content and push it via invoking `send_output`.

calc_best_est_lin_sys(*a_arr*, *a_arr2d*, *x_arr*, *vx_arr2d*, *problim*)
Calculate the best estimate of a linear system **y** = **a** + A * **x**

Additionally determine if the inputs are consistent in view of *problim*.

Parameters

- **a_arr** (*np.ndarray of shape (n_estimates)*) – vector **a** of linear system **y** = **a** + A * **x**
- **a_arr2d** (*np.ndarray of shape (n_estimates, n_sensors)*) – matrix A of linear system **y** = **a** + A * **x**
- **x_arr** (*np.ndarray of shape (n_sensors)*) – vector with sensor values, vector **x** of linear system **y** = **a** + A * **x**
- **vx_arr2d** (*np.ndarray of shape (n_sensors, n_sensors)*) – uncertainty matrix associated with vector x_arr
- **problim** (*float*) – probability limit used for consistency evaluation. Typically 0.95.

Returns

- **isconsist** (*bool*) – indicator whether provided estimates are consistent in view of *problim*
- **ybest** (*float*) – best estimate
- **uybest** (*float*) – standard uncertainty of best estimate
- **chi2obs** (*float*) – observed chi-squared value

static calc_best_estimate(*y_arr*, *vy_arr2d*, *problim*)
Calculate the best estimate for a set of estimates with uncertainties

Additionally determine if the set of estimates are consistent using a provided limit probability.

Parameters

- **y_arr** (*np.ndarray of shape (n)*) – vector of estimates of a measurand Y
- **vy_arr2d** (*np.ndarray of shape (n, n)*) – uncertainty matrix associated with y_arr
- **problim** (*float*) – probability limit used for assessing the consistency of the estimates. Typically, problim equals 0.95.

Returns

- **isconsist** (*bool*) – indicator whether provided estimates are consistent in view of *problim*
- **ybest** (*float*) – best estimate of measurand
- **uybest** (*float*) – uncertainty associated with *ybest*
- **chi2obs** (*float*) – observed value of chi-squared, used for consistency evaluation

static calc_consistent_estimates_no_corr(*y_arr2d*, *uy_arr2d*, *prob_lim*)
Calculation of consistent estimate for sets of estimates y_ij

The y_ij (contained in y_arr2d) are the elements of Y, where each set contains n_estims estimates. The uncertainties are assumed to be independent and given in uy_arr2d. The consistency test is using limit probability limit prob_lim. For each set of estimates, the best estimate, uncertainty, observed chi-2 value and a flag if the provided estimates were consistent given the model are given as output.

Parameters

- **y_arr2d** (*np.ndarray of size (n_rows, n_estimates)*) – each row contains m=n_estimates independent estimates of a measurand
- **uy_arr2d** (*np.ndarray of size (n_rows, n_estimates)*) – each row contains the standard uncertainty u(y_ij) of y_ij = y_arr2d[i,j]
- **prob_lim** (*float*) – limit probability used in consistency test. Typically 0.95.

Returns

- **isconsist_arr** (*bool array of shape (n_rows)*) – indicates for each row if the n_estimates are consistent or not
- **ybest_arr** (*float or np.ndarray of float in shape (n_rows)*) – contains the best estimate for each row of individual estimates
- **uybest_arr** (*float or np.ndarray of float in shape (n_rows)*) – contains the uncertainty associated with each best estimate for each row of *y_arr2d*
- **chi2obs_arr** (*np.ndarray of float in shape (n_rows)*) – observed chi-squared value for each row

calc_lcs(*y_arr*, *vy_arr2d*, *problim*)
Calculate the best estimate of a measurand with associated uncertainty matrix

Parameters

- **y_arr** (*np.ndarray of shape (n)*) – vector with estimates of the measurand
- **vy_arr2d** (*np.ndarray of shape (n, n)*) – uncertainty matrix of the vector y_arr

- **problim** (*float*) – limit probability used in the consistency evaluation. Typically 0.95.

Returns

- **n_solutions** (*int or np.ndarray of ints*) – number of solutions
- **ybest** (*float or np.ndarray of floats*) – best estimate
- **uybest** (*float or np.ndarray of floats*) – standard uncertainty of best estimate
- **chi2obs** (*float or np.ndarray of floats*) – observed chi-squared value
- **indkeep** (*np.ndarray of shape (n) or (n_sols, n)*) – indices of kept estimates

calc_lcss(*a_arr*, *a_arr2d*, *x_arr*, *vx_arr2d*, *problim*)

Calculation of the largest consistent subset of sensor values

Additionally the implied best estimate is returned.

Parameters

- **a_arr** (*np.ndarray of shape (n_estimates)*) – vector **a** of linear system **y** = **a** + A * **x**
- **a_arr2d** (*np.ndarray of shape (n_estimates, n_sensors)*) – matrix A of linear system **y** = **a** + A * **x**
- **x_arr** (*np.ndarray of shape (n_sensors)*) – vector with sensor values vector **x** of linear system **y** = **a** + A * **x**
- **vx_arr2d** (*np.ndarray of shape (n_sensors, n_sensors)*) – uncertainty matrix associated with vector x_arr
- **problim** (*float*) – probability limit used for consistency evaluation. Typically 0.95

Returns

- **n_solutions** (*int or np.ndarray of ints*) – number of solutions
- **isconsist** (*bool or np.ndarray of bool*) – indicator whether provided estimates are consistent in view of *problim*
- **ybest** (*float or np.ndarray of floats*) – best estimate
- **uybest** (*float or np.ndarray of floats*) – standard uncertainty of best estimate
- **chi2obs** (*float or np.ndarray of floats*) – observed chi-squared value

static get_combination(*values*, *n_keep*, *certain_combinations_index*)

Return a certain subset of n_keep elements in a given array

Parameters

- **values** (*np.ndarray*) – original values
- **n_keep** (*int*) – number of elements in subset
- **certain_combinations_index** (*int*) – the index of the desired combination as a result of a call of `combinations(values, n_keep)`

static ind_reduce_a(*a_arr2d*, *epszero*)

Returns the index of a linear dependent row of a matrix A

The motivation for this is, that this row does not contribute any new information to the system.

Parameters

- **a_arr2d** (*np.ndarray*) – The matrix to be reduced as 2-dimensional array

- **epszero** (*float*) – some small constant used for checking equality to zero

Returns the index of the last row that can be taken out

Return type int

init_lcss_parameters(*fsam*, *f1*, *f2*, *ampl_ratio*, *phi1*, *phi2*)
Additional parameters used for this particular example

Provides the prior knowledge needed to make the information contained in the data redundant. This method sets up the vector **a** and matrix *A* for the system **y** = **a** + *A* * **x**.

Parameters

- **fsam** (*float*) – sampling frequency
- **f1** (*float*) – first frequency of interest in signal
- **f2** (*float*) – second frequency of interest in signal
- **ampl_ratio** (*float*) – ratio of the amplitudes of the two frequency components
- **phi1** (*float*) – initial phase of first frequency component
- **phi2** (*float*) – initial phase of second frequency component

init_parameters(*input_data_maxlen: int = 25*, *output_data_maxlen: int = 25*, *sensor_key_list: list = None*, *n_pr: int = 1*, *problim: float = 0.9*, *calc_type: str = 'lcs'*)
Initialize the redundancy agent

Parameters

- **input_data_maxlen** (*int, optional*) – Defaults to 25
- **output_data_maxlen** (*int, optional*) – Defaults to 25
- **sensor_key_list** (*list of str, optional*) – list containing the names of the sensors that should feed data to the Redundancy Agent. Defaults to None
- **n_pr** (*int, optional*) – size of the batch of data that is handled at a time by the Redundancy Agent. Defaults to 1
- **problim** (*float, optional*) – limit probability used for consistency evaluation. Defaults to .9
- **calc_type** (*str, optional*) – calculation type: 'lcs' or 'lcss'. Defaults to 'lcs'

on_received_message(*message*)
Handle incoming data from 'default' channels

Store 'default' data into an internal buffer.

Parameters **message** (*dict*) – Only acceptable channel value is 'default'.

static print_input_lcss(*x_arr*, *vx_arr2d*, *a_arr*, *a_arr2d*, *problim*)
Prints the input parameters of the method

Parameters

- **x_arr** (*np.ndarray of shape (n_sensors)*) – vector with sensor values, vector **x** of linear system **y** = **a** + A * **x**
- **vx_arr2d** (*np.ndarray of shape (n_sensors, n_sensors)*) – uncertainty matrix associated with vector x_arr
- **a_arr** (*np.ndarray of shape (n_estimates)*) – vector **a** of linear system **y** = **a** + A * **x**

- **a_arr2d** (*np.ndarray of shape (n_estimates, n_sensors)*) – matrix A of linear system **y** = **a** + A * **x**
- **problim** (*float*) – probability limit used for consistency evaluation. Typically 0.95

print_output_cbe(*isconsist_arr*, *ybest_arr*, *uybest_arr*, *chi2obs_arr*)
Function to print the full output of calc_best_estimate.

Parameters

- **isconsist_arr** (*bool array of shape (n_rows)*) – indicates for each row if the n_estimates are consistent or not
- **ybest_arr** (*np.ndarray of floats in shape (n_rows)*) – contains the best estimate for each row of individual estimates
- **uybest_arr** (*np.ndarray of floats in shape (n_rows)*) – contains the uncertainty associated with each best estimate for each row of *y_arr2d*
- **chi2obs_arr** (*np.ndarray of floats in shape (n_rows)*) – observed chi-squared value for each row

static print_output_lcs(*n_solutions*, *ybest*, *uybest*, *chi2obs*, *indkeep*, *y_arr*)
Method to print the output of the method *calc_lcs()*

Parameters

- **n_solutions** (*int*) – number of best solutions
- **ybest** (*float or np.ndarray of shape (n_sols)*) – best estimate or vector of best estimates
- **uybest** (*float or np.ndarray of shape (n_sols)*) – standard uncertainty of best estimate or vector with standard uncertainty of best estimates
- **chi2obs** (*float*) – observed chi-squared value of all best solutions
- **indkeep** (*np.ndarray of shape (n) or (n_sols, n)*) – indices of retained estimates of y_arr for the calculation of the best estimate ybest
- **y_arr** (*np.ndarray of shape (n)*) – individual estimates of measurand

static print_output_lcss(*n_solutions*, *ybest*, *uybest*, *chi2obs*, *indkeep*, *x_arr*, *a_arr2d*)
Prints the outputs of the method *calc_lcss()*

Parameters

- **n_solutions** (*int or np.ndarray of ints*) – number of solutions
- **ybest** (*float or np.ndarray of floats*) – best estimate
- **uybest** (*float or np.ndarray of floats*) – standard uncertainty of best estimate
- **chi2obs** (*float or np.ndarray of floats*) – observed chi-squared value
- **indkeep** (*np.ndarray of int*) – indices of kept estimates
- **x_arr** (*np.ndarray of shape (n_estimates)*) – vector **a** of linear system **y** = **a** + A * **x**
- **a_arr2d** (*np.ndarray of shape (n_estimates, n_sensors)*) – matrix A of linear system **y** = **a** + A * **x**

Returns

Return type None

static print_output_single(*isconsist*, *ybest*, *uybest*, *chi2obs*)
Print the output of a single row of the calculate_best_estimate function

Parameters

- **isconsist** (*bool*) – Indicates if provided estimates were consistent
- **ybest** (*float*) – best estimate
- **uybest** (*float*) – uncertainty of best estimate
- **chi2obs** (*float*) – observed value of chi-squared

static reduce_vx(*x_arr*, *vx_arr2d*, *a_arr*, *a_arr2d*, *epszero*)
Reduce the system if matrix Vx is not of full rank

This might be ambiguous, as constant sensor values or offsets have to be estimated and are not known.

Parameters

- **x_arr** (*np.ndarray*) – The vector x to be reduced
- **vx_arr2d** (*np.ndarray*) – The matrix Vx to be reduced as 2-dimensional array
- **a_arr** (*np.ndarray*) – The vector a to be reduced
- **a_arr2d** (*np.ndarray*) – The matrix A to be reduced as 2-dimensional array
- **epszero** (*float*) – some small constant used for checking equality to zero

Returns The reduced matrices and vectors xred_arr, vxred_arr2d, ared_arr, ared_arr2d

Return type np.ndarray, np.ndarray, np.ndarray, np.ndarray

8.5 Metrologically enabled signal agents

class agentMET4FOF.agents.metrological_signal_agents.**MetrologicalGeneratorAgent**(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=N*

An agent streaming a specified signal

Takes samples from an instance of `MetrologicalDataStreamMET4FOF` with sampling frequency `sfreq` and signal frequency `sine_freq` and pushes them sample by sample to connected agents via its output channel.

agent_loop()
Model the agent's behaviour

On state *Running* the agent will extract sample by sample the input datastream's content and push it into its output buffer.

init_parameters(*signal: agentMET4FOF.streams.metrological_base_streams.MetrologicalDataStreamMET4FOF = <agentMET4FOF.streams.metrological_signal_streams.MetrologicalSineGenerator object>*, ***kwargs*)

Initialize the input data stream

Parameters **signal** (MetrologicalDataStreamMET4FOF) – the underlying signal for the generator (defaults to MetrologicalSineGenerator)

CHAPTER 9

agentMET4FOF streams

9.1 Base streams

class agentMET4FOF.streams.base_streams.**DataStreamMET4FOF**

Abstract class for creating datastreams

Data can be fetched sequentially using *next_sample()* or all at once *all_samples()*. This increments the internal sample index *_sample_idx*.

For sensors data, we assume:

- The format shape for 2D data stream (timesteps, n_sensors)
- The format shape for 3D data stream (num_cycles, timesteps , n_sensors)

To create a new DataStreamMET4FOF class, inherit this class and call *set_metadata()* in the constructor. Choose one of two types of datastreams to be created:

- from dataset file (*set_data_source()*), or
- a waveform generator function (*set_generator_function()*).

Alternatively, override the *next_sample()* function if neither option suits the application. For generator functions, *sfreq* is a required variable to be set on *init* which sets the sampling frequency and the time-step which occurs when *next_sample()* is called.

For an example implementation of using generator function, see the built-in `SineGenerator` class. See tutorials for more implementations.

_quantities

Measured quantities such as sensors readings

Type Union[List, DataFrame, np.ndarray]

_target

Target label in the context of machine learning. This can be Remaining Useful Life in predictive maintenance application. Note this can be an unobservable variable in real-time and applies only for validation during offline analysis.

Type Union[List, DataFrame, np.ndarray]

_time
`dtype` can be either `float` or `datetime64` to indicate the time when the *_quantities* were measured.

Type Union[List, DataFrame, np.ndarray]

_current_sample_quantities
Last returned measured quantities from a call to *next_sample()*

Type Union[List, DataFrame, np.ndarray]

_current_sample_target
Last returned target labels from a call to *next_sample()*

Type Union[List, DataFrame, np.ndarray]

_current_sample_time
`dtype` can be either `float` or `datetime64` to indicate the time when the *_current_sample_quantities* were measured.

Type Union[List, DataFrame, np.ndarray]

_sample_idx
Current sample index

Type int

_n_samples
Total number of samples

Type int

_data_source_type
Explicitly account for the data source type: either "function" or "dataset"

Type str

_generator_function
A generator function which takes in at least one argument `time` which will be used in *next_sample()*

Type Callable

_generator_parameters
Any additional keyword arguments to be supplied to the generator function. The generator function call for every sample will be supplied with the `**generator_parameters`.

Type Dict

sfreq
Sampling frequency

Type int

_metadata
The quantities metadata as `time_series_metadata.scheme.MetaData`

Type MetaData

_default_generator_function(*time*)
This is the default generator function used, if non was specified

Parameters **time** (*Union[List, DataFrame, np.ndarray]*) –

_next_sample_data_source(*batch_size: Optional[int] = 1*) → Dict[str, Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray]]

Internal method for fetching latest samples from a dataset

Parameters **batch_size** (*int, optional*) – number of batches to get from data stream, defaults to 1

Returns

latest samples in the form:

```
dict like {
    "quantities": <time series data as a list, np.ndarray or
        pd.Dataframe>,
    "target": <target labels as a list, np.ndarray or pd.Dataframe>,
    "time": <time stamps as a list, np.ndarray or pd.Dataframe of
        float or np.datetime64>
}
```

Return type Dict[str, Union[List, DataFrame, np.ndarray]]

_next_sample_generator(*batch_size: int = 1*) → Dict[str, numpy.ndarray]

Internal method for generating a batch of samples from the generator function.

_set_data_source_type(*dt_type: str = 'function'*)

To explicitly account for the type of data source: either from dataset, or a generator function.

Parameters **dt_type** (*str*) – Either "function" or "dataset"

all_samples() → Dict[str, Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray]]

Return all the samples in the data stream

Returns

all samples in the form:

```
dict like {
    "quantities": <time series data as a list, np.ndarray or
        pd.Dataframe>,
    "target": <target labels as a list, np.ndarray or pd.Dataframe>,
    "time": <time stamps as a list, np.ndarray or pd.Dataframe of
        float or np.datetime64>
}
```

Return type Dict[str, Union[List, DataFrame, np.ndarray]]

has_more_samples() → bool

Tell if there are more samples to extract

next_sample(*batch_size: Optional[int] = 1*) → Dict[str, Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray]]

Fetch the latest samples from the `quantities`, `time` and `target`

Parameters **batch_size** (*int, optional*) – number of batches to get from data stream, defaults to 1

Returns

latest samples in the form:

```
dict like {
    "quantities": <time series data as a list, np.ndarray or
```

(continues on next page)

(continued from previous page)

```
        pd.Dataframe>,
    "target": <target labels as a list, np.ndarray or pd.Dataframe>,
    "time": <time stamps as a list, np.ndarray or pd.Dataframe of
        float or np.datetime64>
}
```

Return type Dict[str, Union[List, DataFrame, np.ndarray]]

randomize_data()
Randomizes the provided quantities, useful in machine learning contexts

reset()
Set the sample count to zero to prepare for new extractions

set_data_source(*quantities: Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray] = None, target: Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray, None] = None, time: Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray, None] = None*)
This sets the data source by providing up to three iterables: `quantities` , `time` and `target` which are assumed to be aligned.

For sensors data, we assume: The format shape for 2D data stream (timesteps, n_sensors) The format shape for 3D data stream (num_cycles, timesteps , n_sensors)

Parameters

- **quantities** (*Union[List, DataFrame, np.ndarray]*) – Measured quantities such as sensors readings.
- **target** (*Optional[Union[List, DataFrame, np.ndarray]]*) – Target label in the context of machine learning. This can be Remaining Useful Life in predictive maintenance application. Note this can be an unobservable variable in real-time and applies only for validation during offline analysis.
- **time** (*Optional[Union[List, DataFrame, np.ndarray]]*) – `dtype` can be either `float` or `datetime64` to indicate the time when the `quantities` were measured.

set_generator_function(*generator_function: Callable = None*, *sfreq: int = None*, ***kwargs*)
Sets the data source to a generator function. By default, this function resorts to a sine wave generator function. Initialisation of the generator's parameters should be done here such as setting the sampling frequency and wave frequency. For setting it with a dataset instead, see *set_data_source()*.

Parameters

- **generator_function** (*Callable*) – A generator function which takes in at least one argument `time` which will be used in *next_sample()*. Parameters of the function can be fixed by providing additional arguments such as the wave frequency.
- **sfreq** (*int*) – Sampling frequency.
- ****kwargs** (*Any*) – Any additional keyword arguments to be supplied to the generator function. The `**kwargs` will be saved as *_generator_parameters*. The generator function call for every sample will be supplied with the `**generator_parameters`.

set_metadata(*device_id: str, time_name: str, time_unit: str, quantity_names: Union[str, Tuple[str, ...]], quantity_units: Union[str, Tuple[str, ...]], misc: Optional[Any] = None*)
Set the quantities metadata as a `MetaData` object

Details you find in the `time_series_metadata.scheme.MetaData` documentation.

Parameters

- **device_id** (*str*) – Name of the represented generator
- **time_name** (*str*) – Name for the time dimension
- **time_unit** (*str*) – Unit for the time
- **quantity_names** (*iterable of str or str*) – A string or an iterable of names of the represented quantities' values
- **quantity_units** (*iterable of str or str*) – An iterable of units for the quantities' values
- **misc** (*Any, optional*) – This parameter can take any additional metadata which will be handed over to the corresponding attribute of the created `Metadata` object

9.2 Signal streams

class agentMET4FOF.streams.signal_streams.**SineGenerator**(*sfreq=500, sine_freq=50, amplitude: float = 1.0, initial_phase: float = 0.0*)

Built-in class of sine wave generator which inherits all methods and attributes from `DataStreamMET4FOF`. *sine_wave_function()* is a custom defined function which has a required keyword `time` as argument and any number of optional additional arguments (e.g `F`) to be supplied to the *DataStreamMET4FOF.set_generator_function()*.

Parameters

- **sfreq** (*int*) – sampling frequency which determines the time step when *next_sample()* is called
- **sine_freq** (*float*) – frequency of wave function
- **amplitude** (*float, optional*) – Amplitude of the wave function. Defaults to 1.
- **initial_phase** (*float, optional*) – Initial phase of the wave function. Defaults to 0.0.

sine_wave_function(*time*, *sine_freq*, *amplitude*, *initial_phase*)

A simple sine wave generator

class agentMET4FOF.streams.signal_streams.**CosineGenerator**(*sfreq=500, cosine_freq=50, amplitude: float = 1, initial_phase: float = 0*)

Built-in class of cosine wave generator which inherits all methods and attributes from `DataStreamMET4FOF`. *cosine_wave_function()* is a custom defined function which has a required keyword `time` as argument and any number of optional additional arguments (e.g `cosine_freq`) to be supplied to the *DataStreamMET4FOF.set_generator_function()*.

Parameters

- **sfreq** (*int*) – sampling frequency which determines the time step when *next_sample()* is called
- **cosine_freq** (*int*) – frequency of wave function
- **amplitude** (*float, optional*) – Amplitude of the wave function. Defaults to 1.0.

- **initial_phase** (*float, optional*) – Initial phase of the wave function. Defaults to 0.0.

cosine_wave_function(*time*, *cosine_freq*, *amplitude*, *initial_phase*)
A simple cosine wave generator

class agentMET4FOF.streams.signal_streams.**StaticSineWithJitterGenerator**(*num_cycles=1000*, *jitter_std=0.02*)

Represents a fixed length sine signal with jitter

Parameters

- **num_cycles** (*int, optional*) – numbers of cycles, determines the signal length by $\pi \cdot num_cycles$, defaults to 1000
- **jitter_std** (*float, optional*) – the standard deviation of the distribution to randomly draw jitter from, defaults to 0.02

9.3 Metrologically enabled base streams

class agentMET4FOF.streams.metrological_base_streams.**MetrologicalDataStreamMET4FOF**(*value_unc: Optional[float] = 0.0*, *time_unc: Optional[float] = 0.0*, *exp_unc: Optional[float] = None*, *cov_factor: Optional[float] = 1.0*)

Abstract class for creating datastreams with metrological information. Inherits from the *DataStreamMET4FOF* class

To create a new *MetrologicalDataStreamMET4FOF* class, inherit this class and call *set_metadata()* in the constructor. Choose one of two types of datastreams to be created:

- from dataset file (*set_data_source()*), or
- a waveform generator function (*set_generator_function()*).

Alternatively, override the *next_sample()* function if neither option suits the application. For generator functions, *sfreq* is a required variable to be set on *init* which sets the sampling frequency and the time-step which occurs when *next_sample()* is called.

For an example implementation of using generator function, see the built-in `MetrologicalSineGenerator` class. See tutorials for more implementations.

_generator_function_unc
A generator function for the time and quantity uncertainties which takes in at least one argument `time` which will be used in *next_sample()*. The return value must be a 2-tuple of time and value uncertainties each of one of the three types:

- np.ndarray
- pandas DataFrame
- list

Type Callable

_uncertainty_parameters
Any additional keyword arguments to be supplied to the generator function. Both the calls of the value generator function and of the uncertainty generator function will be supplied with the `**_uncertainty_parameters`.

Type Dict

_default_uncertainty_generator(*time: Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray], values: Union[List[T], pandas.core.frame.DataFrame, numpy.ndarray]*) → Tuple[numpy.ndarray, numpy.ndarray]
Default (standard) uncertainty generator function

Parameters

- **time** (*Union[List, DataFrame, np.ndarray]*) – timestamps
- **values** (*Union[List, DataFrame, np.ndarray]*) – values corresponding to timestamps

Returns constant time and value uncertainties each of the same shape as `time`

Return type Tuple[np.ndarray, np.ndarray]

_next_sample_generator(*batch_size: int = 1*) → numpy.ndarray
Internal method for generating a batch of samples from the generator function. Overrides *DataStreamMET4FOF._next_sample_generator()*. Adds time uncertainty `ut` and measurement uncertainty `uv` to sample

set_generator_function(*generator_function: Callable = None*, *uncertainty_generator: Callable = None*, *sfreq: int = None*, ***kwargs*) → Callable
Set value and uncertainty generators based on user-defined functions. By default, this function resorts to a sine wave generator function and a constant (zero) uncertainty. Initialisation of the generator's parameters should be done here such as setting the sampling frequency and wave frequency. For setting it with a dataset instead, see *set_data_source()*. Overwrites the default *DataStreamMET4FOF.set_generator_function()* method.

Parameters

- **generator_function** (*callable*) – A generator function which takes in at least one argument `time` which will be used in *next_sample()*.
- **uncertainty_generator** (*callable*) – An uncertainty generator function which takes in at least one argument `time` which will be used in *next_sample()*.
- **sfreq** (*int*) – Sampling frequency.
- ****kwargs** (*Optional[Dict[str, Any]]*) – Any additional keyword arguments to be supplied to the generator function. The `**kwargs` will be

saved as *_uncertainty_parameters*. Both the calls of the value generator function and of the uncertainty generator function will be supplied with the `**uncertainty_parameters`.

Returns The uncertainty generator function

Return type Callable

time_unc
uncertainties associated with timestamps

Type Union[float, Iterable[float]]

value_unc
uncertainties associated with the values

Type Union[float, Iterable[float]]

9.4 Metrologically enabled signal streams

class agentMET4FOF.streams.metrological_signal_streams.**MetrologicalMultiWaveGenerator**(*sfreq: int = 500, freq_a numpy = ar- ray([5 am- pli- tude_a numpy = ar- ray([1 ini- tial_ph numpy = ar- ray([0 in- ter- cept: float = 0, de- vice_id str = 'Mul- ti- Wave- Data- Gen- er- a- tor', time_n str = 'time', time_u str = 's', quan- tity_na Union[Tu- ple[str ...]] = ('Leng 'Mass*

Values with associated uncertainty are returned.

Parameters

- **sfreq** (*float*) – sampling frequency which determines the time step when next_sample is called.
- **intercept** (*float*) – constant intercept of the signal
- **freq_arr** (*np.ndarray of float*) – array with frequencies of components included in the signal
- **amplitude_arr** (*np.ndarray of float*) – array with amplitudes of components included in the signal
- **initial_phase_arr** (*np.ndarray of float*) – array with initial phases of components included in the signal
- **noisy** (*bool*) – boolean to determine whether the generated signal should be noisy or "clean" defaults to True

class agentMET4FOF.streams.metrological_signal_streams.**MetrologicalSineGenerator**(*sfreq: int = 500, sine_freq: float = 50, amplitude: float = 1.0, initial_phase: float = 0.0, device_id: str = 'SineGenerator', time_name: str = 'time', time_unit: str = 's', quantity_names: Union[str, Tuple[str, ...]] = 'Voltage', quantity_units: Union[str, Tuple[str, ...]] = 'V', misc: Optional[Any] = 'Simple*

Built-in class of sine wave generator

Parameters

- **sfreq** (*int, optional*) – Sampling frequency which determines the time step when *next_sample()* is called. Defaults to 500.
- **sine_freq** (*float, optional*) – Frequency of the wave function. Defaults to 50.0.
- **amplitude** (*float, optional*) – Amplitude of the wave function. Defaults to 1.0.
- **initial_phase** (*float, optional*) – Initial phase of the wave function. Defaults to 0.0.
- **device_id** (*str, optional*) – Name of the represented generator. Defaults to 'Sine-Generator'.
- **time_name** (*str, optional*) – Name for the time dimension. Defaults to 'time'.
- **time_unit** (*str, optional*) – Unit for the time. Defaults to 's'.
- **quantity_names** (*iterable of str or str, optional*) – An iterable of names of the represented quantities' values. Defaults to ('Voltage')
- **quantity_units** (*iterable of str or str, optional*) – An iterable of units for the quantities' values. Defaults to ('V')
- **misc** (*Any, optional*) – This parameter can take any additional metadata which will be handed over to the corresponding attribute of the created `Metadata` object. Defaults to 'Simple sine wave generator'.
- **value_unc** (*iterable of floats or float, optional*) – standard uncertainty(ies) of the quantity values. Defaults to 0.1.
- **time_unc** (*iterable of floats or float, optional*) – standard uncertainty of the time stamps. Defaults to 0.0.

_sine_wave_function(*time*, *sine_freq*, *amplitude*, *initial_phase*)

A simple sine wave generator

CHAPTER 10

agentMET4FOF dashboard

class agentMET4FOF.dashboard.Dashboard.**AgentDashboard**(*dashboard_modules=[]*, *dashboard_layouts=[]*, *dashboard_update_interval=3*, *max_monitors=10*, *ip_addr='0.0.0.0'*, *port=8050*, *agentNetwork='127.0.0.1'*, *agent_ip_addr=3333*, *agent_port=None*, *network_stylesheet=[]*, *hide_default_edge=True*, ***kwargs*)

Class for the web dashboard which runs with the AgentNetwork object, which by default are on the same IP. Optional to run the dashboard on a separate IP by providing the right parameters. See example for an implementation of a separate run of dashboard to connect to an existing agent network. If there is no existing agent network, error will show up. An internal _Dashboard_Control object is instantiated inside this object, which manages access to the AgentNetwork.

_show_startup_message()

This method prints the startup message of the webserver/dashboard

init_app_layout(*update_interval_seconds=3*, *max_monitors=10*, *dashboard_layouts=[]*, *network_stylesheet=[]*, *hide_default_edge=True*, ***kwargs*)

Initialises the overall dash app "layout" which has two sub-pages (Agent network and ML experiment)

Parameters

- **update_interval_seconds** (*float or int*) – Auto refresh rate which the app queries the states of Agent Network to update the graphs and display
- **max_monitors** (*int*) – Due to complexity in managing and instantiating dynamic figures, a maximum number of monitors is specified first and only the each Monitor Agent will occupy one of these figures. It is not ideal, but will undergo changes for the better.

Returns app

Return type Dash app object

is_port_available(*ip_addr*, *_port*)
Check if desired ip and port are available.

run()
This is actually executed on calling start() and brings up the server

class agentMET4FOF.dashboard.Dashboard.**AgentDashboardProcess**(*dashboard_modules=[]*, *dashboard_layouts=[]*, *dashboard_update_interval=3*, *max_monitors=10*, *ip_addr='0.0.0.0'*, *port=8050*, *agentNetwork='127.0.0.1'*, *agent_ip_addr=3333*, *agent_port=None*, *network_stylesheet=[]*, *hide_default_edge=True*, ***kwargs*)
Represents an agent dashboard for the osBrain backend

terminate()
This is shutting down the application server serving the web interface

class agentMET4FOF.dashboard.Dashboard.**AgentDashboardThread**(*dashboard_modules=[]*, *dashboard_layouts=[]*, *dashboard_update_interval=3*, *max_monitors=10*, *ip_addr='127.0.0.1'*, *port=8050*, *agentNetwork='127.0.0.1'*, *agent_ip_addr=3333*, *agent_port=None*, ***kwargs*)
Represents an agent dashboard for the Mesa backend

run()
This is actually executed on calling start() and brings up the server

terminate()
This is shutting down the application server serving the web interface

CHAPTER 11

agentMET4FOF agent network

class agentMET4FOF.network.**AgentNetwork**(*ip_addr='0.0.0.0', port=3333, connect=False, log_filename='log_file.csv', dashboard_modules=True, dashboard_extensions=[], dashboard_update_interval=3, dashboard_max_monitors=10, dashboard_port=8050, backend='osbrain', mesa_update_interval=0.1, network_stylesheet=[{'selector': 'node', 'style': {'label': 'data(id)', 'shape': 'rectangle', 'text-valign': 'center', 'text-halign': 'center', 'color': '#FFF', 'text-outline-width': 1.5, 'text-outline-color': '#000232'}}, {'selector': 'edge', 'style': {'curve-style': 'unbundled-bezier', 'mid-target-arrow-shape': 'triangle', 'arrow-scale': 2, 'line-color': '#4287f5', 'mid-target-arrow-color': '#4287f5', 'label': 'data(channel)', 'text-outline-width': 1.5, 'text-outline-color': '#000232', 'color': '#FFF', 'text-margin-x': '10px', 'text-margin-y': '20px'}}, {'selector': '.rectangle', 'style': {'shape': 'rectangle'}}, {'selector': '.triangle', 'style': {'shape': 'triangle'}}, {'selector': '.octagon', 'style': {'shape': 'octagon'}}, {'selector': '.ellipse', 'style': {'shape': 'ellipse'}}, {'selector': '.bluebackground', 'style': {'background-color': '#c4fdff'}}, {'selector': '.blue', 'style': {'background-color': '#006db5'}}, {'selector': '.coalition', 'style': {'background-color': '#c4fdff', 'text-valign': 'top', 'text-halign': 'center'}}, {'selector': '.coalition-edge', 'style': {'line-style': 'dashed'}}, {'selector': '.outline', 'style': {'color': '#fff', 'text-outline-color': '#888', 'text-outline-width': 2}}], **dashboard_kwargs*)

Object for starting a new Agent Network or connect to an existing Agent Network

An existing Agent Network can be specified by ip & port. Provides function to add agents, (un)bind agents, query agent network state, set global agent states Interfaces with an internal _AgentController which is hidden from user.

class Coalition(*name='Coalition'*, *agents=[]*)
A special class for grouping agents.

It is rendered as a parent group on the dashboard, along with its member agents.

class MesaModel
A MESA Model

shutdown()
Shutdown entire MESA model with all agents and schedulers

step()
Advance the model by one step.

class _AgentController(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)
Unique internal agent to provide control to other agents

Automatically instantiated when starting server. Provides global control to all agents in network.

_get_logger()
Internal method to access the Logger relative to the nameserver

_transform_string_into_valid_name(*name: str*) → str
Ensure that name does not contain invalid characters

osBrain does not allow spaces in agents' names, so we replace them by underscores.
Parameters **name** (*str*) – a string that is supposed to be an agent's name for assigning it or to search for
Returns the cleaned version of the name, i.e. for `backend == 'osbrain'` without spaces
Return type str

add_coalition(*new_coalition*)
Instantiates a coalition of agents

add_coalition_agent(*name: str, agents: List[Union[agentMET4FOF.agents.base_agents.AgentMET4FOF, osbrain.proxy.Proxy]]*)
Add agents into the coalition

agents(*exclude_names: Optional[List[str]] = None*) → List[str]
Returns all or subset of agents' names connected to agent network

For the osBrain backend , the mandatory agents `AgentController`, `Logger` are never returned.
Parameters **exclude_names** (*str, optional*) – if present, only those names are returned which contain `exclude_names`'s value
Returns requested names of agents
Return type list[str]

del_coalition()
Delete all coalitions

get_agent(*agent_name: str*) → Union[agentMET4FOF.agents.base_agents.AgentMET4FOF, osbrain.proxy.Proxy, None]
Returns a particular agent connected to Agent Network
Parameters **agent_name** (*str*) – Name of agent to search for in the network
Returns The particular agent with the provided name or None, if no agent with the provided name can be found

Return type Union[*AgentMET4FOF*, Proxy]

get_coalition(*name: str*)
Gets the coalition based on provided name

init_parameters(*ns=None*, *backend='osbrain'*, *mesa_model=None*, *log_mode=True*)
User provided function to initialize parameters of choice.

remove_agent_from_coalition(*coalition_name: str*, *agent_name: str*)
Remove agent from a coalition

class _Logger(*name=''*, *host=None*, *serializer=None*, *transport=None*, *attributes=None*, *backend='osbrain'*, *mesa_model=None*)
An internal logger agent which is instantiated with each AgentNetwork

It collects all the logs which are sent to it, and print them and optionally save them into a csv log file. Since the user is not expected to directly access the logger agent, its initialisation option and interface are provided via the AgentNetwork object.

When log_info of any agent is called, the agent will send the data to the logger agent.

init_parameters(*log_filename='log_file.csv'*, *save_logfile=True*)
User provided function to initialize parameters of choice.

_get_controller() → agentMET4FOF.network.AgentNetwork._AgentController
Internal method to access the AgentController relative to the nameserver

_get_controller_mode()
Internal method to get mode of agent controller

Returns **state** – State of Agent Network

Return type str

_get_logger() → agentMET4FOF.network.AgentNetwork._Logger
Internal method to access the Logger relative to the nameserver

_set_controller_mode(*state: str*)
Internal method to set mode of agent controller

Parameters **state** (*str*) – State of agent controller to set

add_agent(*name=''*, *agentType: Type[agentMET4FOF.agents.base_agents.AgentMET4FOF] = <class 'agentMET4FOF.agents.base_agents.AgentMET4FOF'>*, *log_mode=True*, *buffer_size=1000*, *ip_addr=None*, *loop_wait=None*, ***kwargs*)
Instantiates a new agent in the network.

Parameters

- **name** (*str, optional*) – Unique name of agent, defaults to the agent's class name.
- **agentType** (*Type[AgentMET4FOF] or subclass of AgentMET4FOF, optional*) – Agent class to be instantiated in the network. Defaults to `AgentMET4FOF`
- **log_mode** (*bool, optional*) – Determines if messages will be logged to background Logger Agent. Defaults to `True`.

Returns Newly instantiated agent

Return type *AgentMET4FOF*

add_coalition(*name='Coalition_1'*, *agents=[]*)
Instantiates a coalition of agents.

add_coalition_agent(*name='Coalition_1'*, *agents=[]*)
Add agents into the coalition

agents (*filter_agent: Optional[str] = None*) → List[str]
Returns all or subset of agents' names connected to agent network

Parameters **filter_agent** (*str, optional*) – if present, only those names are returned which contain `filter_agent`'s value

Returns requested names of agents

Return type list[str]

agents_by_type (*only_type: Optional[Type[agentMET4FOF.agents.base_agents.AgentMET4FOF]] = <class 'agentMET4FOF.agents.base_agents.AgentMET4FOF'>*) → Set[Union[agentMET4FOF.agents.base_agents.AgentMET4FOF, osbrain.proxy.Proxy, None]]
Returns all or a subset of agents connected to an agent network

As expected, the returned set might be empty, if there is no agent of the specified class present in the network.

Parameters **only_type** (*Type[AgentMET4FOF], optional*) – if present, only those agents which are instances of the class `only_type` or a subclasses are listed

Returns requested agents' objects depending on the backend either instances of subclasses of `AgentMET4FOF` or of osBrain's''Proxy''.

Return type Set[*AgentMET4FOF*, Proxy]

bind_agents (*source*, *target*, *channel='default'*)
Binds two agents' communication channel in a unidirectional manner

Any subsequent calls of *source.send_output()* will reach *target* agent's message queue.

Parameters

- **source** (AgentMET4FOF) – Source agent whose Output channel will be binded to *target*
- **target** (AgentMET4FOF) – Target agent whose Input channel will be binded to *source*

connect (*ip_addr: Optional[str] = '127.0.0.1'*, *port: Optional[int] = 3333*)
Connects to an existing agent network's name server for osBrain backend

Parameters

- **ip_addr** (*str, optional*) – IP Address of osBrain name server to connect to, defaults to "127.0.0.1"
- **port** (*int, optional*) – Port of osBrain name server to connect to, defaults to 3333

get_agent (*agent_name: str*) → Union[agentMET4FOF.agents.base_agents.AgentMET4FOF, osbrain.proxy.Proxy, None]
Returns a particular agent connected to Agent Network

Parameters **agent_name** (*str*) – Name of agent to search for in the network

Returns The particular agent with the provided name or None, if no agent with the provided name can be found

Return type Union[*AgentMET4FOF*, Proxy]

get_coalition (*name*)
Returns the coalition with the provided name

remove_agent (*agent*)
Reset all agents' states and parameters to their initialization state

remove_coalition_agent(*coalition_name*, *agent_name=''*)
Remove agent from coalition

reset_agents()
Reset all agents' states and parameters to their initialization state

set_agents_state(*filter_agent: Optional[str] = None*, *state: Optional[str] = 'Idle'*)
Blanket operation on all agents to set their `current_state` to given state

Can be used to define different states of operation such as "Running", "Idle, "Stop", etc.. Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*.

Parameters

- **filter_agent** (*str, optional*) – Filter name of agents to set the states
- **state** (*str, optional*) – State of agents to set

set_running_state(*filter_agent: Optional[str] = None*)
Blanket operation on all agents to set their `current_state` to "Running"

Parameters **filter_agent** (*str, optional*) – Filter name of agents to set the states

set_stop_state(*filter_agent=None*)
Blanket operation on all agents to set their *current_state* attribute to "Stop"

Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*.

Parameters **filter_agent** (*str*) – (Optional) Filter name of agents to set the states

shutdown()
Shuts down the entire agent network and all agents

start_server_mesa()
Starts a new AgentNetwork for Mesa

start_server_osbrain(*ip_addr: Optional[str] = '127.0.0.1'*, *port: Optional[int] = 3333*)
Starts a new agent network's name server for osBrain

Parameters

- **ip_addr** (*str, optional*) – IP Address of osBrain name server to start, defaults to "127.0.0.1"
- **port** (*int, optional*) – Port of osBrain name server to start, defaults to 3333

unbind_agents(*source*, *target*)
Unbinds two agents communication channel in a unidirectional manner

This is the reverse of *bind_agents()*

Parameters

- **source** (AgentMET4FOF) – Source agent whose Output channel will be unbinded from *target*
- **target** (AgentMET4FOF) – Target agent whose Input channel will be unbinded from *source*

CHAPTER 12

agentMET4FOF utilities

12.1 Buffering for agents

This module contains the buffer classes utilized by the agents

It contains the following classes:

- *AgentBuffer*: Buffer class which is instantiated in every agent to store data incrementally
- *MetrologicalAgentBuffer*: Buffer class which is instantiated in every metrological agent to store data

class agentMET4FOF.utils.buffer.**AgentBuffer**(*buffer_size: int = 1000*)

Buffer class which is instantiated in every agent to store data incrementally

This buffer is necessary to handle multiple inputs coming from agents.

We can access the buffer like a dict with exposed functions such as .values(), .keys() and .items(). The actual dict object is stored in the variable *buffer*.

buffer

The buffer can be a dict of iterables, or a dict of dict of iterables for nested named data. The keys are the names of agents.

Type dict of iterables or dict of dicts of iterables

buffer_size

The total number of elements to be stored in the agent *buffer*

supported_datatypes

List of all types supported and thus properly handled by the buffer. Defaults to `np.ndarray`, list and Pandas `DataFrame`

Type list of types

_concatenate(*iterable: Union[numpy.ndarray, list, pandas.core.frame.DataFrame], data: Union[numpy.ndarray, list, pandas.core.frame.DataFrame], concat_axis: int = 0*) → Iterable[T_co]

Concatenate the given `iterable` with `data`

Handles the concatenation function depending on the datatype, and truncates it if the buffer is filled to *buffer_size*.

Parameters

- **iterable** (*any in supported_datatype*) – The current buffer to be concatenated with.
- **data** (*np.ndarray, DataFrame, list*) – New incoming data

Returns the original buffer with the data appended

Return type any in supported_datatype

_iterable_filled(*iterable: Sized*) → Optional[bool]

Internal method for checking on length of iterables of supported types

Parameters **iterable** (*Any*) – Expected to be an iterable of one of the supported datatypes but could be any.

Returns True if the iterable is of one of the supported datatypes and has reached *buffer_size* in length or False if not or None in case it is not of one of the supported datatypes.

Return type bool or None

static _popleft(*iterable: Union[numpy.ndarray, list, pandas.core.frame.DataFrame], n: Optional[int] = 1*) → Tuple[Union[numpy.ndarray, list, pandas.core.frame.DataFrame], Union[numpy.ndarray, list, pandas.core.frame.DataFrame]]

Internal handler of the actual popping mechanism based on type of iterable

Parameters

- **n** (*int*) – Number of elements to retrieve from buffer.
- **iterable** (any in *supported_datatypes*) – The current buffer to retrieve from.

Returns

- 2-tuple of each either one of `np.ndarray`,
- list or Pandas `DataFrame` – The retrieved elements and the residual items in the buffer

buffer_filled(*agent_from: Optional[str] = None*) → bool

Checks whether buffer is filled, by comparing against the *buffer_size*

For nested dict, this returns True if any of the iterables is beyond the *buffer_size*. For any of the dict values , which is not one of *supported_datatypes* this returns None.

Parameters **agent_from** (*str, optional*) – Name of input agent in the buffer dict to be looked up. If `agent_from` is not provided, we check for all iterables in the buffer (default).

Returns True if either the or any of the iterables has reached *buffer_size* or None in case none of the values is of one of the supported datatypes. False if all present iterable can take at least one more element according to *buffer_size*.

Return type bool or None

check_supported_datatype(*obj: object*) → bool

Checks whether *value* is an object of one of the supported data types

Parameters **obj** (*object*) – Value to be checked

Returns **result** – True if value is an object of one of the supported data types, False if not

Return type boolean

clear(*agent_from: Optional[str] = None*)
Clears the data in the buffer

Parameters **agent_from** (*str, optional*) – Name of agent, if `agent_from` is not given, the entire buffer is flushed. (default)

items()
Interface to access the internal dict's items()

keys()
Interface to access the internal dict's keys()

popleft(*n: Optional[int] = 1*) → Union[Dict[KT, VT], numpy.ndarray, list, pandas.core.frame.DataFrame]
Pops the first n entries in the buffer

Parameters **n** (*int*) – Number of elements to retrieve from buffer

Returns

- dict, `np.ndarray`, list or Pandas
- `DataFrame` – The retrieved elements

store(*agent_from: Union[Dict[str, Union[numpy.ndarray, list, pandas.core.frame.DataFrame]], str], data: Union[numpy.ndarray, list, pandas.core.frame.DataFrame, float, int] = None, concat_axis: Optional[int] = 0*)
Stores data into *buffer* with the received message

Checks if sender agent has sent any message before. If it did, then append, otherwise create new entry for it.

Parameters

- **agent_from** (*dict | str*) – if type is dict, we expect it to be the agentMET4FOF dict message to be compliant with older code (keys `from` and `data` present'), otherwise we expect it to be name of agent sender and `data` will need to be passed as parameter
- **data** (*np.ndarray, DataFrame, list, float or int*) – Not used if `agent_from` is a dict. Otherwise `data` is compulsory.
- **concat_axis** (*int, optional*) – axis to concatenate on with the buffering for numpy arrays. Default is 0.

update(*agent_from: Union[Dict[str, Union[numpy.ndarray, list, pandas.core.frame.DataFrame]], str], data: Union[numpy.ndarray, list, pandas.core.frame.DataFrame, float, int] = None*)
Overrides data in the buffer dict keyed by `agent_from` with value `data`

If `data` is a single value, this converts it into a list first before storing in the buffer dict.

Parameters

- **agent_from** (*str*) – Name of agent sender
- **data** (*np.ndarray, DataFrame, list, float or int*) – New incoming data

values()
Interface to access the internal dict's values()

class agentMET4FOF.utils.buffer.**MetrologicalAgentBuffer**(*buffer_size: int = 1000*)
Buffer class which is instantiated in every metrological agent to store data

This buffer is necessary to handle multiple inputs coming from agents.

We can access the buffer like a dict with exposed functions such as .values(), .keys() and .items(). The actual dict object is stored in the attribute `buffer`. The list in `supported_datatypes` contains one more element for metrological agents, namely `TimeSeriesBuffer`.

_concatenate(*iterable: time_series_buffer.buffer.TimeSeriesBuffer, data: Union[numpy.ndarray, list, pandas.core.frame.DataFrame], concat_axis: int = 0*) → time_series_buffer.buffer.TimeSeriesBuffer

Concatenate the given `TimeSeriesBuffer` with `data`

Add `data` to the `TimeSeriesBuffer` object.

Parameters

- **iterable** (*TimeSeriesBuffer*) – The current buffer to be concatenated with.
- **data** (*np.ndarray, DataFrame, list*) – New incoming data

Returns the original buffer with the data appended

Return type TimeSeriesBuffer

convert_single_to_tsbuffer(*single_data: Union[List[T], Tuple, numpy.ndarray]*)

Convert common data in agentMET4FOF to `TimeSeriesBuffer`

Parameters **single_data** (*iterable of iterables (list, tuple, np.ndarray) with shape (N, M)*) –

- M==2 (pairs): assumed to be like (time, value)
- M==3 (triple): assumed to be like (time, value, value_unc)
- M==4 (4-tuple): assumed to be like (time, time_unc, value, value_unc)

Returns the new `TimeSeriesBuffer` object

Return type TimeSeriesBuffer

update(*agent_from: str, data: Union[Dict[KT, VT], List[T], Tuple, numpy.ndarray]*) → time_series_buffer.buffer.TimeSeriesBuffer

Overrides data in the buffer dict keyed by *agent_from* with value *data*

Parameters

- **agent_from** (*str*) – Name of agent sender
- **data** (*dict or iterable of iterables (list, tuple, np.ndarray) with shape (N, M)*) – the data to be stored in the metrological buffer

Returns the updated `TimeSeriesBuffer` object

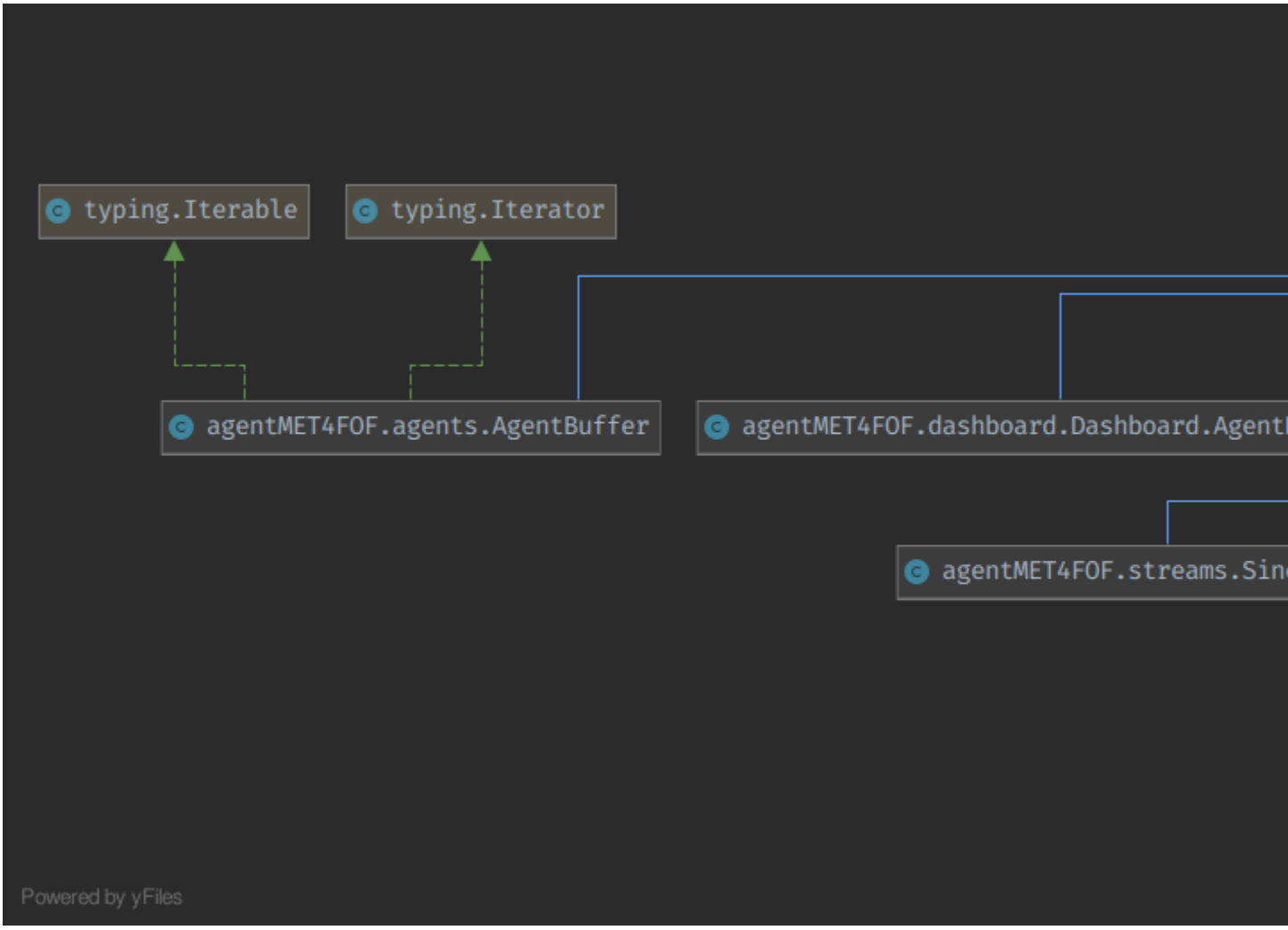
Return type TimeSeriesBuffer

CHAPTER 13

UML diagrams of agentMET4FOF

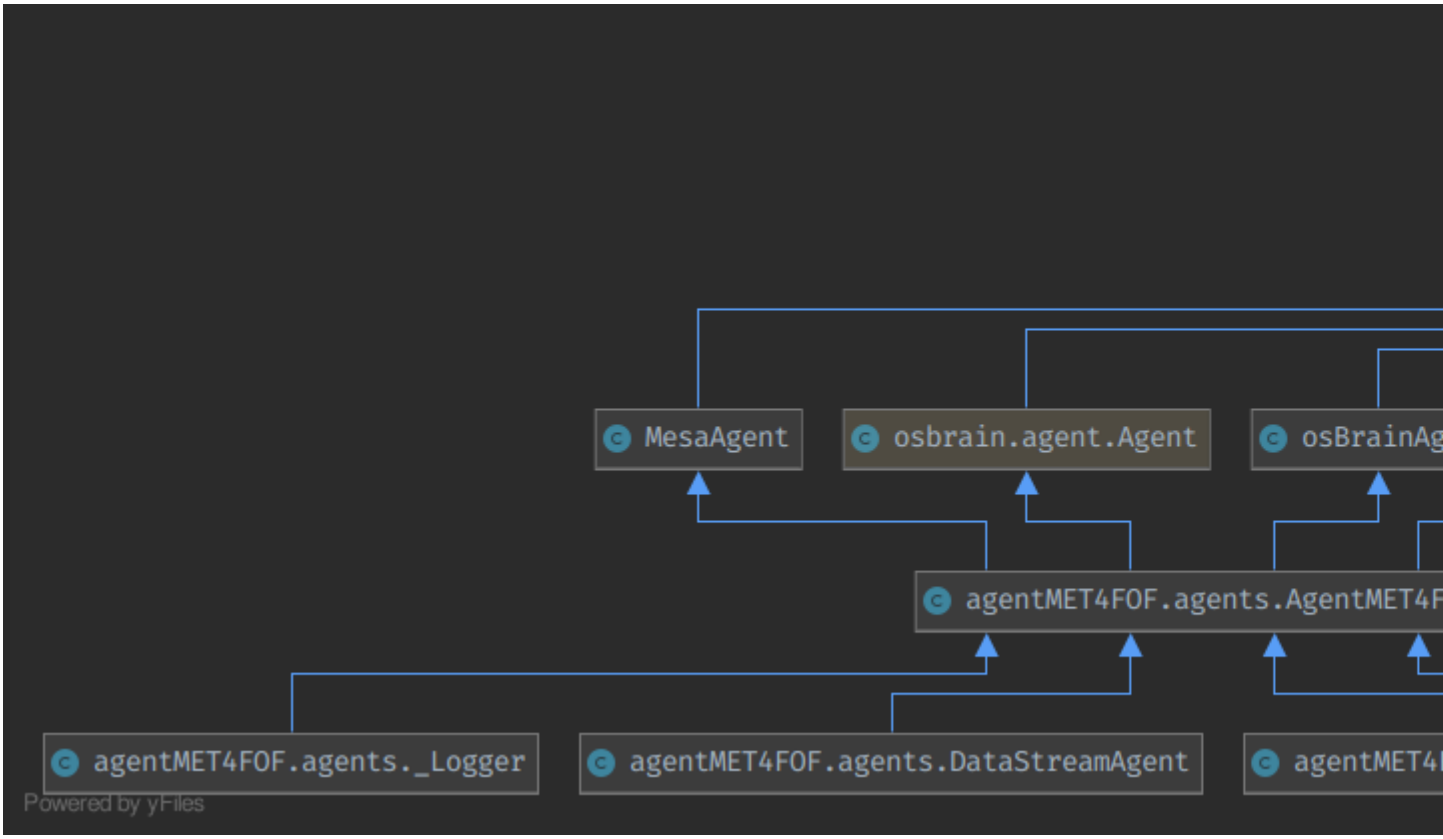
In this section UML class diagrams of almost all components of agentMET4FOF are listed and shown. The images are click- and zoomable in the browser but can be downloaded for further investigation via right-click.

13.1 Overview



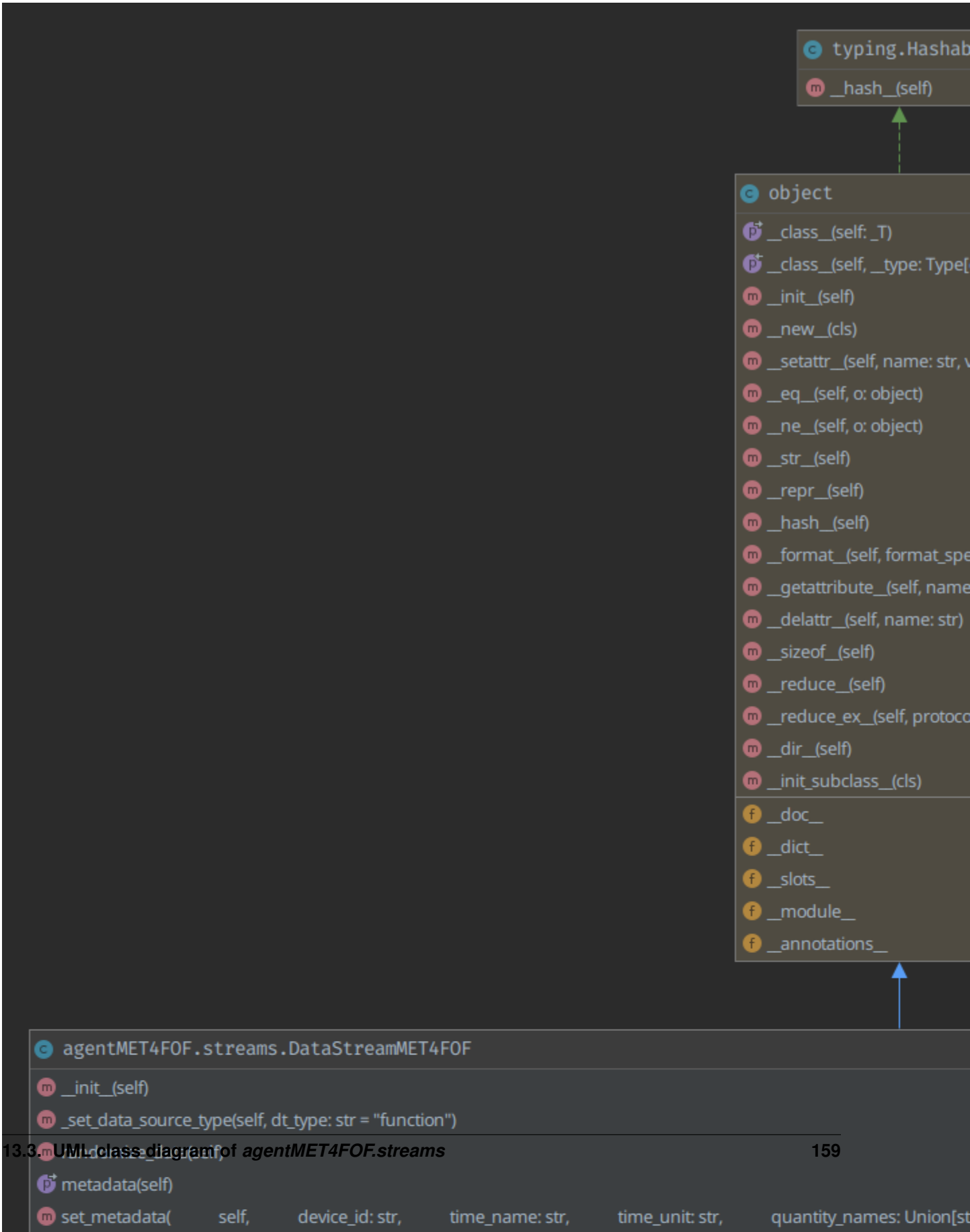
of all agentMET4FOF classes

13.2 UML class diagram of *agentMET4FOF.agents*

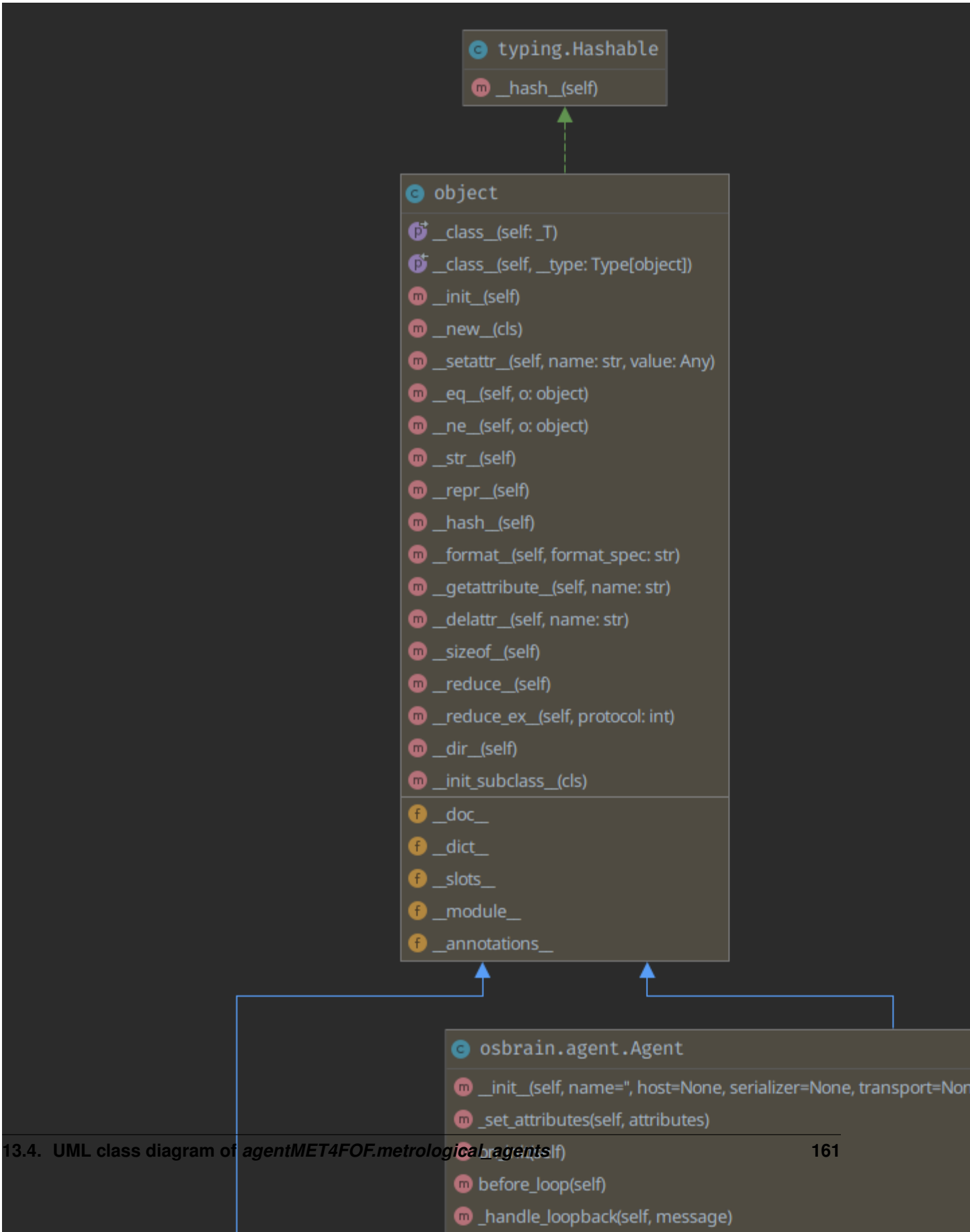


of classes in agentMETFOF.agents

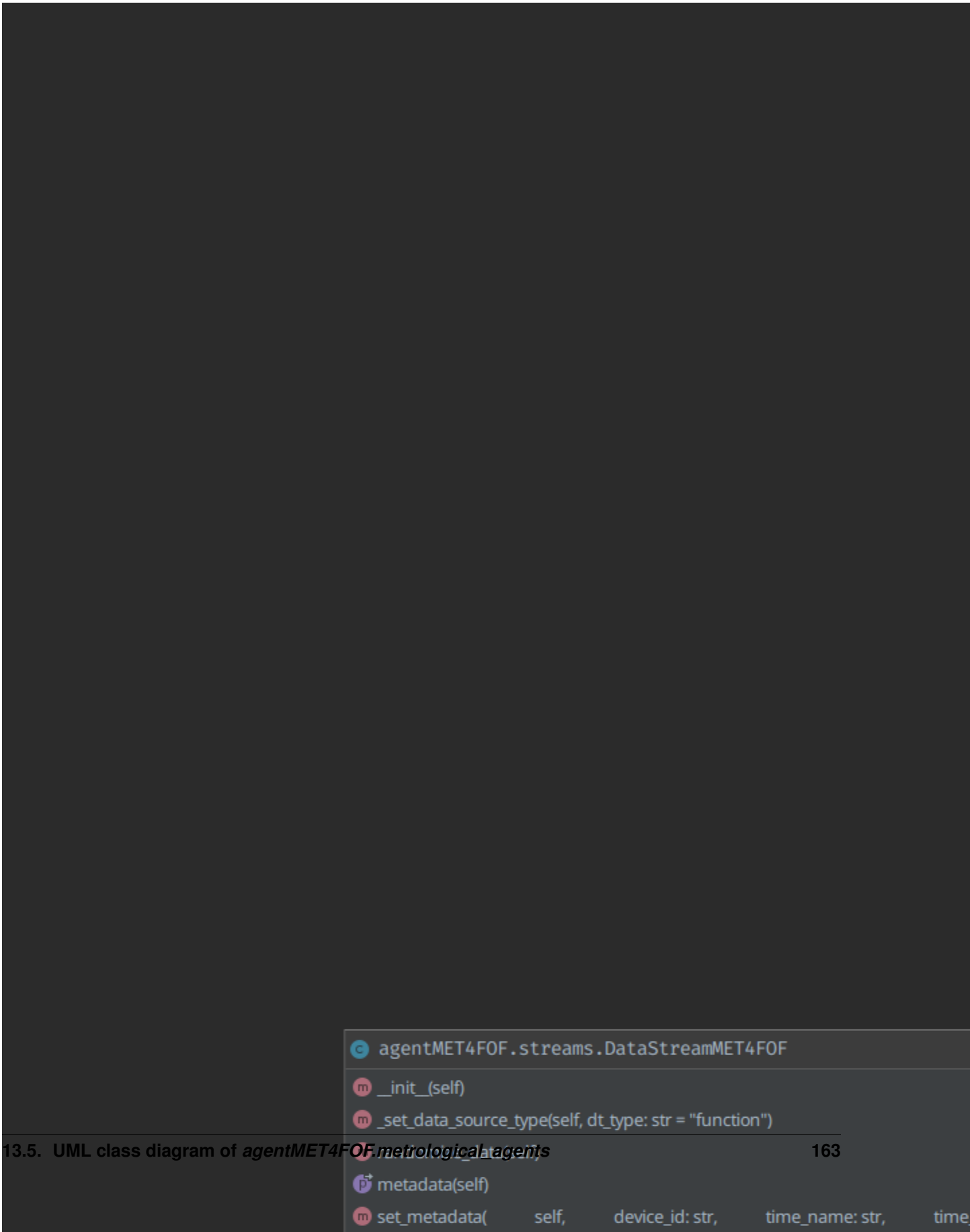
13.3 UML class diagram of *agentMET4FOF.streams*



13.4 UML class diagram of *agentMET4FOF.metrological_agents*



13.5 UML class diagram of *agentMET4FOF.metrological_agents*



13.6 Creation of the UMLs

We used *PyCharm Professional* to create the diagrams in "Hierarchic Group Layout".

Date of creation for all diagrams: 2021-02-03 on commit 54a4863.

CHAPTER 14

Indices and search:

- genindex
- modindex
- search

CHAPTER 15

References:

Bibliography

[Bang2019] Bang X. Yong, A. Brintrup Multi Agent System for Machine Learning Under Uncertainty in Cyber Physical Manufacturing System, 9th Workshop on Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future

[Kok2020_1] G. Kok, P. Harris "Uncertainty Evaluation for Metrologically Redundant Industrial Sensor Networks", 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 2020, pp. 84-88, doi: 10.1109/MetroInd4.0IoT48571.2020.9138297.

[Kok2020_2] G. Kok, P. Harris "Quantifying Metrological Redundancy in an Industry 4.0 Environment", 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, Roma, Italy, 2020, pp. 464-468, doi: 10.1109/MetroInd4.0IoT48571.2020.9138235.

Python Module Index

a

Index

Symbols

A

B

C

D

G

H

I

K

L

M

N

O

P

R

S

T

U

V