LTB Manual

Release 1.0.0

Jinning Wang

Apr 29, 2023

CONTENTS:

Useful Links: Website | YouTube Channel | Report Issues

The CURENT Large-scale Testbed (LTB) is a state-of-the-art research facility designed for rapid prototyping of power systems. It is a tightly integrated, closed-loop system consisting of four major independent sub-packages: ANDES for dynamic simulation, AMS (under development) for market simulation, DiME for distributed messaging environment, and AGVis for grid visualization. These LTB packages can be used individually or in a federated manner, making it a versatile tool for power system research and development.

	LTB	ANDES	AMS	AGVis	DiME
Documentation	LTB Doc	ANDES Doc	AMS Doc	AGVis Doc	DiME Doc
Repository	LTB Repo	ANDES Repo	AMS Repo	AGVis Repo	DiME Repo
Q&A	LTB Q&A	ANDES Q&A	AMS Q&A	AGVis Q&A	DiME Q&A

Getting started

New to CURENT LTB? Check out the getting started guides. These tutorials will take you through the process of installing LTB and running your first simulation.

To the getting started guides

Gallery

Get a deeper understanding of the comprehensive power system simulation capabilities of LTB by exploring our gallery of example simulations and visualizations.

To the gallery

Using LTB for Research?

Please cite our paper [Li2020] [Cui2021] [Parsly2022] if LTB is used in your research for publication.

CHAPTER

ONE

GETTING STARTED

1.1 Platform Overview

The CURENT Large-scale Testbed (LTB) is a state-of-the-art research facility designed for rapid prototyping of power systems. It is a tightly integrated, closed-loop system consisting of four major independent sub-packages: ANDES for dynamic simulation, AMS (under development) for market simulation,DiME for distributed messaging environment, and AGVis for grid visualization. These LTB packages can be used individually or in a federated manner, making it a versatile tool for power system research and development.

LTB is developed and actively maintained by the CURENT LTB Team. To get involved,

- Report issues in the GitHub issues page
- Learn version control with the command-line git or GitHub Desktop

This work was supported in part by the Engineering Research Center Program of the National Science Foundation and the Department of Energy under NSF Award Number EEC-1041877 and the CURENT Industry Partnership Program.

1.2 Installation

1.2.1 New to Python

Setting Up Mambaforge

If you are new to Python and want to get started quickly, you can use Mambaforge, which is a conda-like package manager configured with conda-forge.

Step 1:

Downloaded the latest Mambaforge for your platform from https://github.com/conda-forge/miniforge#mambaforge. Most users will use `x86_64(amd64)` for Intel and AMD processors. Mac users with Apple Silicon should use `arm64(Apple Silicon)` for best performance.

Next, complete the Mambaforge installation on your system.

Note: Mambaforge is a drop-in replacement for conda. If you have an existing conda installation, you can replace all following `mamba` commands with `conda` and achieve the same functionality.

If you are using Anaconda or Miniconda on Windows, you should open `Anaconda Prompt` instead of `Miniforge Prompt`.

Step 2:

Open Terminal (on Linux or maxOS) or *Miniforge Prompt* (on Windows, **not cmd!!**). Make sure you are in a conda environment - you should see `(base)` prepended to the command-line prompt, such as `(base) C:\Users\username>`.

Create an environment for LTB (recommended)

```
mamba create --name ltb python=3.8
```

Activate the new environment with

```
mamba activate ltb
```

Note: You will need to activate the `ltb` environment every time in a new Miniforge Prompt or shell.

If these steps complete without error, you now have a working Python environment. See the commands at the top to *Getting started* LTB.

1.2.2 Extra support package

Some LTB features require extra support packages, which are not installed by default. For example, to build the documentation, one will need to install development packages. Other packages will be required for interoperability.

The extra support packages are specified in groups. The following group names are supported, with descriptions given below:

- `dev`: packages to support development such as testing and documentation

Note: Extra support packages are not supported by conda/mamba installation. One needs to install LTB with `pip`.

To install packages in the `dev` when installing LTB, do:

```
pip install ltb[dev]
```

To install all extra packages, do:

```
pip install ltb[all]
```

One can also inspect the `requirements-extra.txt` to identify the packages for manual installation.

1.2.3 Develop Install

The development mode installation is for users who want to modify the code and, for example, develop new models or routines. The benefit of development mode installation is that changes to source code will be reflected immediately without re-installation.

Step 1: Get LTB source code

As a developer, you are strongly encouraged to clone the source code using `git` from either your fork or the original repository. Clone the repository with

```
git clone https://github.com/CURENT/ltb
```

Note: Replace the URL with yours to use your fork. With `git`, you can later easily update the source code and perform version control.

Alternatively, you can download the LTB source code from https://github.com/CURENT/ltb and extract all files to the path of your choice. Although works, this method is discouraged because tracking changes and pushing back code edits will require significant manual efforts.

Step 2: Install dependencies

In the Mambaforge environment, use `cd` to change directory to the LTB root folder. The folder should contain the `setup.py` file.

Install dependencies with

```
mamba install --file requirements.txt
mamba install --file requirements-extra.txt
```

Alternatively, you can install them with `pip`:

```
pip install -r requirements.txt
pip install -r requirements-extra.txt
```

Step 3: Install LTB in the development mode using

```
python3 -m pip install -e .
```

Note the dot at the end. Pip will take care of the rest.

Note: The LTB version number shown in `pip list` will stuck at the version that was intalled, unless LTB is develop-installed again. It will not update automatically with `git pull`.

To check the latest version number, check the preamble by running the `ltb` command or chek the output of `python -c "import ltb; print(ltb.__version__)"`

Note: LTB updates may infrequently introduce new package requirements. If you see an `ImportError` after updating LTB, you can manually install the missing dependencies or redo *Step 2*.

Note: To install extra support packages, one can append `[NAME_OF_EXTRA]` to `pip install -e .`. For example, `pip install -e .[dev]` will install packages to support documentation when installing LTB in the development, editable mode.

1.2.4 Updating LTB

Warning: If LTB has been installed in the development mode using source code, you will need to use `git` or the manual approach to update the source code. In this case, Do not proceed with the following steps, as they will install a separate site-package installation on top of the development one.

Regular LTB updates will be pushed to both `conda-forge` and Python package index. It is recommended to use the latest version for bug fixes and new features. We also recommended you to check the *Release notes* before updating to stay informed of changes that might break your downstream code.

Depending you how you installed LTB, you will use one of the following ways to upgrade.

If you installed it from mamba or conda, run

```
conda install -c conda-forge --yes ltb
```

If you install it from PyPI (namely, through `pip`), run

```
python3 -m pip install --yes ltb
```

1.3 Tutorial

LTB can be used as a command-line tool or a library. The command-line interface (CLI) comes handy to run studies. As a library, it can be used interactively in the IPython shell or the Jupyter Notebook. This chapter describes the most common usages.

1.3.1 Command line

LTB is invoked from the command line using the command `ltb`

```
██╗  ████████╗██████╗   | CURENT Large-scale Testbed Platform
██║  ╚══██╔══╝██╔══██╗  | Version 0.post46.dev0+ge095432
██║     ██║   ██████╔╝  | ANDES 1.8.6; AMS 0.6.1.post8+g93f012d;
██║     ██║   ██╔══██╗  | AGVis 3.0.0.post91+g81b1a89; DiME;
███████╗██║   ██████╔╝  | Python 3.9.16 on Linux, 04/27/2023 02:02:31 AM
╚══════╝╚═╝   ╚═════╝   | This program comes with ABSOLUTELY NO WARRANTY.

usage: ltb [-h] [-v {1,10,20,30,40}] {run,misc,doc,plot,selftest,st,demo} ...

positional arguments:
{run,misc,doc,plot,selftest,st,demo}
                        [demo] run gallery;

optional arguments:
-h, --help            show this help message and exit
-v {1,10,20,30,40}, --verbose {1,10,20,30,40}
                        Verbosity level in 10-DEBUG, 20-INFO, 30-WARNING, or␣
↪40-ERROR.
```

1.3.2 Jupyter Notebook

You can use LTB in the Jupyter Notebook to create interactive simulations and visualizations, as well as to analyze and plot data generated by LTB simulations. Check out the LTB gallery (comming soon) for examples and tutorials on how to use LTB in the Jupyter Notebook.

1.4 License

1.4.1 GNU Public License v3

Copyright 2023 CURENT LTB Team

LTB is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

LTB is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

1.5 Quick install

Before LTB comes to conda and pip, you can install it from source.

Step 1: Get LTB source code, with all submodules

```
git clone https://github.com/CURENT/ltb.git --recursive
```

Note: When you clone LTB from the Git repository with suffix `--recursive`, the submodules are cloned as well.

Replace the URL with yours to use your fork. With `git`, you can later easily update the source code and perform version control.

Step 2: Install dependencies

In the Mambaforge environment, use `cd` to change directory to the LTB root folder. The folder should contain the `setup.py` file.

Install dependencies with `pip`:

```
pip install -r requirements.txt
pip install -r requirements-extra.txt
```

Step 4: Install LTB in the development mode using

```
python3 -m pip install -e .
```

Note the dot at the end. Pip will take care of the rest.

Note: The versions of each submodule that LTB uses are specified in the `requirements.txt` file. If you require a different version of a submodule, you can manually install a specified version using `pip install`.

Alternatively, if you want to use the latest version of each submodule, you can install it manually using `pip install git+https://github.com/CURENT/andes.git@develop` or other package source.

CHAPTER

TWO

GALLERY

The LTB gallery is a collection of example simulations and visualizations that demonstrate the range and depth of LTB's capabilities in power system simulation. These examples showcase LTB's features and functionality, including dynamic simulations, market analysis, dispath-dynamic co-simulations, among others. Explore the gallery to get a better understanding of what LTB can do and how it can help you in your power system research.

Check out our collection of demo videos on YouTube.

More examples coming soon! Welcome contributions from the community. Submit your own examples to the gallery and check back often for updates.

2.1 Dynamic Simulation

Dynamic simulation is a fundamental aspect of modeling and analyzing the behavior of power systems under transient conditions. It provides critical insights into the power systems transient behavior.

LTB ANDES is a comprehensive dynamic simulation tool that provides a rich library of power system components, rapid modeling capabilities, flexible extendability, and high performance.

To get started with dynamic simulation in LTB, check out the ANDES examples for tutorials of how to use ANDES to create dynamic simulations of power systems.

2.2 Geo-visualization

Geo-visualization is a powerful tool for exploring and visualizing large-scale power systems. LTB provides a range of tools and libraries for creating interactive maps and visualizations of power systems, including the AGVis visualization tool.

AGVis is a web-based visualization tool for exploring and analyzing large-scale power systems. It allows users to create custom visualizations of power system topology, power flows, and other system characteristics. AGVis provides an interactive map interface that allows you to zoom in and out, pan, and explore different aspects of the power system topology. It also includes support for a range of visualization techniques, including heatmaps, time series plots, and scatter plots, among others.

Together with other tools and libraries available in LTB, AGVis provides a powerful set of tools for exploring and visualizing large-scale power systems using LTB.

CHAPTER THREE

RELEASE NOTES

3.1 v1.0 Notes

3.1.1 v1.0.0 (2023-04-29)

- Initial release LTB series tools, including ANDES, DiME, and AGVis.
- Improve documentation for LTB.
- Update README.

3.2 Our Story So Far [Update 2023-04]

In 2014-2015, the director of CURENT, Dr. Kevin Tomsovic, proposed the idea of creating a closed-loop testbed for power systems research. He appointed Dr. Fran Li to lead the development work of what is now known as the CURENT Large-scale Testbed (LTB) or CLTB.

During the first two years of development, the team built models of the North American power grid, including WECC and reduced EI operation and planning models, with various renewable energy penetration scenarios. They also created a nationwide HVDC overlay and integrated initial versions of a visualizer and data transfer package called DiME into the CLTB platform.

In 2017, the team decided to develop their own dynamic simulation engine called ANDES, which was created by Dr. Hantao Cui and has become a cornerstone package in CLTB. ANDES can now be used as a standalone dynamic simulation tool as well.

Around the same time, newer versions of DiME and the visualizer (now called AGVis) were developed with contributions from Nick West, Nicholas Parsly, and Jinning Wang.

CHAPTER

FOUR

API REFERENCE

4.1 Others

BIBLIOGRAPHY

[Li2020] F. Li, K. Tomsovic and H. Cui, "A Large-Scale Testbed as a Virtual Power Grid: For Closed-Loop Controls in Research and Testing," in IEEE Power and Energy Magazine, vol. 18, no. 2, pp. 60-68, March-April 2020, doi: 10.1109/MPE.2019.2959054.

[Cui2021] H. Cui, F. Li and K. Tomsovic, "Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis," in IEEE Transactions on Power Systems, vol. 36, no. 2, pp. 1373-1384, March 2021, doi: 10.1109/TPWRS.2020.3017019.

[Parsly2022] N. Parsly, J. Wang, N. West, Q. Zhang, H. Cui and F. Li, "DiME and AGVIS: A Distributed Messaging Environment and Geographical Visualizer for Large-scale Power System Simulation," in arXiv, Nov. 2022, doi: arXiv.2211.11990.