HiPAS GridLAB-D
Performance Analysis CPR Report

David P. Chassin and Alyona I. Teyber
SLAC National Accelerator Laboratory
Menlo Park, California (USA)

Deliverable 4.1.3 - Performance Analysis CPR Report

15 November 2022

Table of Contents

This report was produced with funding from grant EPC-17-046 by the California Energy Commission. SLAC National Accelerator Laboratory is operated by Stanford University for the US Department of Energy under Contract DE-AC02-76SF00515.

Introduction

GridLAB-D is a research simulator developed by the US Department of Energy (DOE) and used to study future electricity distribution systems that include advanced grid technology, emerging customer end-uses, and distributed energy resources such as solar panel, energy storage, and electric vehicle chargers. HiPAS GridLAB-D is a commercial-grade high-performance version of the DOE software that is developed for four important analysis use-cases for California, i.e., load electrification, hosting capacity analysis, tariff analysis, and distribution system resilience analysis. These four use-cases address California climate change mitigation goals, including electricity infrastructure decarbonization, as well as the response to climate change impacts on electricity distribution system infrastructure, such as wildfire planning and operations.

HiPAS GridLAB-D is delivered as an open-source product available free-of-charge for users, and easily installed on the most widely used computing platforms, including cloud systems, on premise servers, desktop workstations, and laptop computers. The software support and maintenance infrastructure is open to contributors, and administered by a team of professional software engineers, in consultation with a highly experienced team of electrical and mechanical engineers.

HiPAS GridLAB-D has been adopted by the Linux Foundation Energy, an open source foundation focused on the power systems sector, and hosted within the Linux Foundation to provide a neutral, collaborative community to build the shared digital investments that are transforming the world's relationship to energy.

This report documents the testing and validation methodology and results for the HiPAS GridLAB-D system, including the main source code, the application modules, subcommands, tools, Python language interface, object libraries, use-case templates, weather data, benchmarking framework, data and model format converters, and tutorial examples.

Methodology

The supporting repositories for HiPAS GridLAB-D and its support infrastructure comprises a total of ten repositories which disseminate code, data, and models needed to run the four principal use-cases. Each repository provides its own independent automated testing and validation framework to ensure the releases work according to the specification and meet the requirements of the use-cases. Table 1 summarizes the repositories, content, and testing/validation approach used.

Table 1: HiPAS GridLAB-D Testing/Validation Infrastructure

Repository Name	Content Description	Testing/Validation Method
gridlabd	Simulator code and tools	gridlabd --validate
gridlabd-template	Use-case templates	Autotest script
gridlabd-benchmarks	Benchmark models	Manual tests
gridlabd-weather	Weather data	Autotest script
gridlabd-models	GridLAB-D reference models	Autotest script
gridlabd-library	GridLAB-D object libraries	Autotest script
gridlabd-examples	Tutorial examples	Manual tests

GridLAB-D Validate Process

GridLAB-D uses a built-in validation system to verify the test and validate the main simulation source repository. The validation process searches the entire code tree to folders named `autotest` and runs every model with the name pattern `test_*.glm`. If the test model file name includes the string `_err` then the test must fail--this is used to ensure that the system is capable of detecting failures as well as ensure success.

The result of each test is stored in a subfolder, and if the autotest fails (or succeeds when it should fail) then all the intermediate and output files are saved by the test framework for later download and diagnostics.

The validation process is run on every push or pull request to the `develop` and `master` branch. A pull request that fails validation cannot be merged to the target branch and will remain blocked until the validation test passes or one of the administrators overrides the block. This ensures the integrity of the master releases.

Installation download of the *gridlabd* simulator distribution images are tested manually when they are generated after the *master* branch is updated.

Autotest script

Most support repositories run a script called `autotest` to test and validate the contents of the repository whenever a push or a pull request to `master` or `develop` branches occurs. The `autotest` script report how many tests were completed, how many passed, and how many failed.

A pull request that has one or more failed autotests cannot be merged to the target branch and will remain blocked until the autotest passes with 100% success or one of the administrators overrides the block. This ensures the integrity of the master releases.

Manual test

Manual testing is required for repositories that are infrequently updated or too complex to automatically test. The main README files of these repositories contain the guidelines for manual testing.

Implementation

All automated testing and validation is performed using GitHub Actions. The standard action workflows update only on push and pull requests to *master* and *develop* branches on code repositories, and *master* branches only on data repositories. All workflows use the *hipas/gridlabd:latest* docker image for validation, except the code base in *gridlabd*, which builds its own image to validate itself. The validation image build process typically requires around 5 minutes to complete on GitHub.

Successful workflows generate a "passing" badge, which is displayed in the respective repository's `README.md` file, as well as in the main code base's `README.md`.

Failed workflows generate a `autotest-results.tarz` file for download to assist in diagnostics. This files is preserved for 90 days before it is deleted. Failed workflows also generated a "failed" badge, which is displayed in the respective repository's `README.md` file, as well as in the main code base's `README.md`.

gridlabd

The `gridlabd` repository uses GitHub actions to perform 724 GridLAB-D validation tests on the components listed in Table 2. This validation process typically requires about 7 minutes to complete on GitHub.

The validation tests for each function or class answer the following questions.

1. Does the function or class work as documented?
2. Does the function of class detect invalid usage?
3. Does the function of class detect invalid input data?
4. Does the function or class produce valid output data?
5. Does the function or class detect when it produces invalid output data?

The taxonomy feeders (*autotest*), *converter*, *geodata* and most *module* tests include numerical validation tests to ensure that correct values are obtained. In general *assert* tests are used within the model to verify that numerical results are within required limits, e.g., voltages, currents, and powers are within prescribed ranges. Of the 724 tests in the repository 322 of these include numerical validation tests using the *assert* test framework, as shown in the "Assert Tests" column of Table 2.

Table 2: GridLAB-D component validation tests

Component Name	Component Description	Total Tests	Assert Tests
autotest	Taxonomy feeder models	46	46
converters	Data and model format converters	15	5
geodata	Geographic data processing	1 (8 units)	1 (8 units)
module/assert	Model validation classes	15	15
module/climate	Climate classes	4	3
module/commercial	Commercial building classes	38	25
module/generators	Electricity generators classes	23	22
module/industrial	Industrial load classes	2	0
module/influxdb	Influx database classes	1 (see Note 1)	0
module/market	Retail market classes	68	66
module/mysql	MySQL database classes	8 (see Note 1)	0
module/optimize	Optimizer classes	3	3
module/powerflow	Powerflow solver classes	260	241
module/reliability	Reliability analysis classes	9	7
module/residential	Residential building classes	95	81
module/resilience	Resilience analysis classes	2	0
module/revenue	Revenue analysis classes	1	0
module/tape	CSV file input/output classes	25	4
python	Python interface	1	(See Note 2)
source	GridLAB-D core	141	26
subcommands	GridLAB-D subcommands	17	(See Note 2)
tools	Tools	8	(See Note 2)

Notes:

1. Tests must be performed manually due to special runtime access requirements.
2. Assert tests cannot be performed on external code bases using the GridLAB-D validation system.

gridlabd-weather

The `gridlabd-weather` repository uses GitHub actions to validate the 1020 TMY3 weather files in the `US` folder. Each weather file is loaded into GridLAB-D to ensure that the weather data is complete and valid. This test process typically requires about 5 minutes to complete on GitHub.

gridlabd-benchmarks

The `gridlabd-benchmarks` repository is used to perform performance tests on the North American Taxonomy feeder models for various analyses. The feeder models are listed in Table 3.

Currently, only the ICA use-case is benchmarked using this framework. The benchmark feeder models are obtained from the DOE Modern Grid Initiative Taxonomy of Northwest America Feeders[1].

gridlabd-models

The `gridlabd-models` repository contains the authoritative test models used to validate the powerflow solvers. Three groups of test models are included, with a total 40 models, as shown Table 3.

Table 3 - GridLAB-D powerflow test models

IEEE Standard Tests	PG&E Taxonomy	DOE Taxonomy
13.glm 37.glm 123.glm 342.glm 8500.glm	AL0001.glm AT0001.glm BR0015.glm BU0001.glm D0001.glm HL0004.glm MC0001.glm MC0006.glm MO0001.glm OC0001.glm PL0001.glm TMP0009.glm	R1-12470-1.glm R1-12470-2.glm R1-12470-3.glm R1-12470-4.glm R1-25000-5.glm R2-12470-1.glm R2-12470-2.glm R2-12470-3.glm R2-25000-4.glm R2-35000-5.glm R3-12470-1.glm R3-12470-2.glm R3-12470-3.glm R4-12470-1.glm R4-12470-2.glm R4-25000-3.glm R5-12470-1.glm R5-12470-2.glm R5-12470-3.glm R5-12470-4.glm R5-12470-5.glm R5-25000-6.glm R5-35000-7.glm

The autotest script verifies that the test models compile and solve.

gridlabd-library

The `gridlabd-library` repository contains commonly used configuration libraries such as cables, poles, transformers, and voltage regulators. The libraries are grouped by country, region, and organization.

The validation test loads, compiles, and initializes all the libraries found in the repository. After this process is completed, the library is saved in JSON format to complete the test.

[1] https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-18035.pdf

Results

The HiPAS GridLAB-D testing and validation results are presented in the main repository README.md file, as shown in Figure 1. The code repositories for the main simulation and templates report the validation status for both the release (*master*) and development (*develop*) versions. All the other repositories report the validation status of the *master* only.

Repository	Status
GridLAB-D	master passing develop passing
Templates	master passing develop passing
Weather	validate passing
Library	validate passing
Models	validate passing
Benchmarks	Manual test (see README.md)
Examples	Manual test (see README.md)

Figure 1: HiPAS GridLAB-D system status (source: https://code.gridlabd.us)

gridlabd

The simulation validation completes with 100% success. This is a condition for approval of a release of `develop` branches to `master`, as well as a condition for merging pull requests into `develop` branches. The typical validation test time is approximately 7 minutes on GitHub.

gridlabd-weather

The weather data validation completes 1020 weather file checks with 100% success, as required. The typical validation test time is approximately 5 minutes on GitHub.

gridlabd-library

The library data validation completes 2 library files checks with 100% success, as required. The typical validation test time is less than 1 second on GitHub.

gridlabd-models

The reference model validation completes 40 model checks with 100% success, as required.

gridlabd-template

The template validation completes 2 templates (ICA and loadfactor) with 100% success. Three additional templates are still pending completion before final validation. These include electrification, tariff design, and resilience analysis.

gridlabd-benchmarks

The ICA analysis performance is benchmarked relative to a model in which a single power flow solution is completed. The performance trend for ICA depends most strongly on the number of network size (i.e., number of nodes) and the number of DER units in the model. The ICA solution time in seconds as a function of network size is given by

$$t \simeq 0.05\, n^2$$

where n is the number of nodes. The ICA solution time in seconds as a function of the number of DER units in the model is given by

$$t \simeq 0.6\, n + 3.0$$

The results for the individual taxonomy feeders are shown in Table 4.

Table 4: Benchmark performance for ICA analysis

Feeder	Node Count	Link Count	DER Count	Solver Time (s)	ICA Time (s)
R1-12470-1	660	1832	20	3.4	9.4
R1-12470-2	336	852	13	1.1	1.5
R1-12470-3	96	75	21	0.8	1.0
R1-12470-4	331	392	12	1.0	1.4
R1-25000-1	506	496	90	1.0	10.7
R2-12470-1	654	751	82	1.2	18.1
R2-12470-2	274	644	8	1.2	1.7
R2-12470-3	855	1812	11	1.3	1.8

Table 4: Benchmark performance for ICA analysis

Feeder	Node Count	Link Count	DER Count	Solver Time (s)	ICA Time (s)
R2-25000-1	474	800	72	1.8	32.7
R2-35000-1	1952	1811	447	2.0	185.1
R3-12470-1	764	1513	64	1.9	27.1
R3-12470-2	392	328	62	1.3	22.6
R3-12470-3	2220	5358	108	5.2	179.2
R4-12470-1	725	1599	75	1.5	11.5
R4-12470-2	316	645	21	1.0	1.8
R4-25000-1	234	511	1	1.0	1.0
R5-12470-1	366	684	48	1.2	9.5
R5-12470-2	411	638	46	1.6	22.5
R5-12470-3	1839	4042	182	6.3	293.6
R5-12470-4	473	1056	32	2.6	25.0
R5-25000-1	986	1706	14	2.6	20.2
R5-35000-1	434	769	47	1.8	20.2

Use Case Validation

In addition to the various solvers and converters implemented in HiPAS GridLAB-D, four use cases were deployed as templates. These templates require numerical validation to ensure the methodology produces correct results. Many of the tests described above include such tests. This section describes additional tests performed on the use-case templates to ensure numerical validity.

Integration Capacity Analysis (ICA)

The GLOW development team tested the ICA methodology by comparing the results on the IEEE 123 test model with those obtained from Cyme's ICA implementation. They report achieving "very similar" results. No further data on these tests is available to the SLAC team.

Electrification

TODO: provide a summary of the validation test results obtained.

Tariff Design

TODO: provide a summary of the validation test results obtained.

Performance of tariff design template was completed and results obtained are as follows:

- Verified using manual calculation for PG&E residential tariffs while testing 1 house simulation for a year. Percent error determined: 0%
- Performance verified using Docker environment release `hipas/gridlabd:develop`: 499.10s user 13.97s system 99% cpu 8:34.16 total
-

Resilience

TODO: provide a summary of the validation test results obtained.

Conclusions

The HiPAS GridLAB-S testing and validation system provides an automated framework for verifying the functionality of the GridLAB-D code as well as the support code and data repositories. GridLAB-D code and use-case template releases are required to pass the validation tests with 100% success. Development versions are used to stage code submissions and verify their integration using the same testing and validation system. Data libraries are also validated to ensure their usability and correctness. The overall status of the entire testing and validation suite is reported in real-time using the GitHub actions badging system and displayed in the main code base's README.md file.

The testing and validation framework will continue to be developed and improved upon a commercialization and technology transfer is completed. Most of the manual validation processed will eventually be automated. In addition, code coverage and syntax checks such as "lint" will be deployed to improve the process.