HiPAS GridLAB-D
Final Report Implementation Addendum

Learning-Accelerated Powerflow Solver

Quasi-static time series (QSTS) power flow simulators such as HiPAS GridLAB-D are often applied to use-cases that may involve simulating a power system over a long simulation horizon at high temporal resolution or over numerous simulation scenarios. These simulations often require performing a large number of sequential power flow calculations, which can be a computational bottleneck. Accelerating the computational speed of such simulations is important for enabling more types of simulation analyses.

Various approaches have been proposed in literature for accelerating power flow simulation, such as algorithmic improvements to power flow solvers and using power flow approximations that are faster to solve. One approach that has recently received attention is using machine learning or other data-driven models to replace the standard power flow solver. Such models can be trained on a dataset of solutions and used to predict the solution to new power flow problems. Using such models reduces computation time since they are generally much faster to evaluate than running a standard power flow solver. However, such approaches are generally sensitive to the size and composition of the training dataset and the hyperparameters of the learned model, and may not have robust performance when applied to new data.

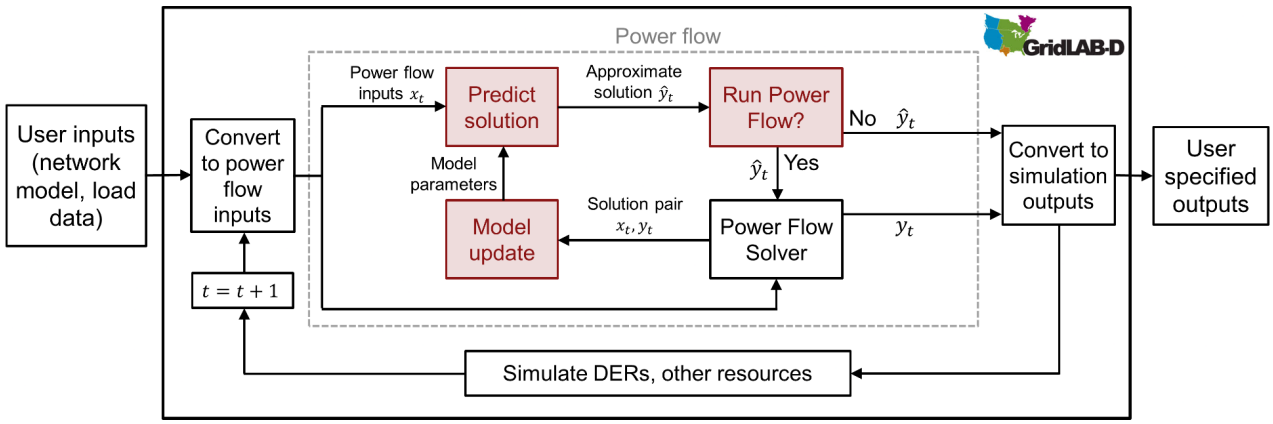
The HiPAS GridLAB-D project investigated the use of different data-driven power flow approximations for accelerating QSTS power flow simulations. The final framework that was developed was implemented in HiPAS GridLAB-D. This approach was designed to (i) accelerate simulation time, (ii) provide a level of accuracy in the power flow solutions that is acceptable for the identified HiPAS GridLAB-D use-cases, and (iii) be robust to simulation inputs and hyperparameter selection. The method is designed specifically for QSTS simulation, which involves sequential static power flow calculations where the power flow inputs often do not vary significantly from one timestep to another. Avoiding performing these redundant calculations can help reduce the overall computation time of the simulation.

The architecture of the proposed method is shown in Figure 1. A data-driven model is trained and updated in-the-loop with the standard power flow solver, and the data-driven model can be used to bypass the standard solver when appropriate. At each timestep, a data-driven model is used to predict an approximate power flow solution

given a set of power flow inputs. Then, a decision is made whether or not to use the approximate solution as the final solution or to instead run the standard power flow solver to obtain a "ground truth" solution. This decision can be made using an appropriate heuristic, which can be based on variables such as previous prediction errors or the number of timesteps such the power flow solver was last run. When the standard power flow solver is run, it is initialized with the approximate solution. After the solver converges, the parameters of the data-driven model are updated based on the new solution.

This approach reduces computation time by: (i) reducing the total number of required power flow computations and (ii) improving the convergence of the standard power flow solver by warm-starting it with an approximate solution. In the HiPAS GridLAB-D implementation, a linear regression model is used for the data-driven prediction. The features of the data-driven model are a function of the time-varying powerflow inputs, which consist mostly of the non-zero real and reactive power injections. The parameters are updated using a recursive least squares filter with exponential forgetting, and separate sets of parameters are maintained for each topology state in the simulation. Online updating of the approximate powerflow model during the simulation allows the parameters to adapt to changes in the power flow inputs (e.g., daily, weekly, or seasonal load changes, new topology states). Currently, the HiPAS GridLAB-D implementation of this framework is only compatible with the Newton Raphson-based powerflow solver.

Figure 1: Architecture of learning-accelerated power flow framework



Converters

The automatic converter suite in HiPAS GridLAB-D is implemented as a two-stage process, with stage-one components handling file formats, such as CSV, MDB, JSON, and GLM, and stage-two components handling the file semantics for each format, such as CSV AMI, GLM objects, etc.

The stage-one converters are selected based on the files' input and output extensions. For example, converting "config.csv" to "config.glm" will select the stage-one converter "csv2glm" to handle the file.

The stage-two converters are selected based on the "type" and "format" specification on the command line of the converter, such as "ami" to "ceus", to convert from AMI data to a CEUS load model.

Table 1 provides a list of the stage-one converters supported by HiPAS GridLAB-D. Appendix 3 provides a list of the stage-two converters supported by each stage-one converter listed in Table 1.

Table 1: Stage-one Converters

Input format	Output format	Converter name	Stage-two
CSV	GLM	csv2glm	See Table C-1
CSV	KML	csv-geodata2kml	GeoData CSV only
GLM	OMD	glm2omd	N/A
GZ	GLM	gz2glm	N/A
JSON	CSV	json2csv	See Table C-2
JSON	MD	json2md	N/A
JSON	PNG	json2png	See Table C-3
JSON	TXT	json2txt	N/A
JSON	ZIP	json2zip	N/A
MDB	GLM	mdb2glm	See Table C-4
OMD	GLM	omd2glm	N/A
TMY3	GLM	tmy32glm	N/A
TXT	GLM	txt2glm	See Table C-5

Input format	Output format	Converter name	Stage-two
XLS	CSV	xls2csv	See Table C-6
ZIP	GLM	zip2glm	N/A

Stage-one file converters support input and output file formats, and use the command line arguments "--type" and "--format" to identify the stage-two converter process to use.

Geodata

HiPAS GridLAB-D introduced the "geodata" subcommand to assist in processing geographic information system (GIS) data used in GridLAB-D models. Geodata is delivered in CSV format with columns to identify the various types of data associated with locations, and rows to identify each location at which data is available.

There are two basic geodata commands, one to create a geodata table from GIS records, and the second to merge additional geodata into an existing geodata table. The following GIS records may be merged into an existing geodata table.

- **Address**: address resolution using the Nominatim system is available to convert between US Postal Service addresses and geographic locations specified using latitude and longitude and added to the geodata table.
- **Census**: census boundary data from the US Department of Commerce TIGER data is available to obtain the boundaries for various census regions and added to the geodata table.
- **Distance**: distances between geographic locations as specified using latitude and longitude are calculated and added to the geodata table.
- **Elevation**: ground elevation obtained from the US geological survey is added to the geodata table.
- **Firerisk**: fire risk data from the US Forest Service is added to the geodata table.
- **Powerline**: powerline sag and sway are added to the geodata table.
- **Utility**: utility service territory for geographic locations is added to the geodata table.
- **Vegetation**: vegetation data from the California Forest Observatory is added to the geodata table.

- **Weather**: weather data from NREL and NOAA is added to the geodata table.

Subcommands and Tools

HiPAS GridLAB-D introduced a subcommand handling capability for command-line parsing. Any keyword provided immediately after the executable name and found listed among the valid subcommand will be executed using all the remaining command line arguments. The following subcommands are available:

- **assert**: verifies that an expression is true.
- **aws**: provide access to AWS command.
- **check**: verifies that GLM or JSON file is a valid GridLAB-D model.
- **compare**: compares two GridLAB-D models.
- **contributors**: provides a list of contributors to HiPAS GridLAB-D.
- **convert**: converts a file from one format/semantics to another.
- **geodata**: creates or merges geographic datasets.
- **git**: provides access to GIT commands.
- **help**: provides help on GridLAB-D subcommands and tools.
- **job**: provides access to GridLAB-D job control.
- **json-get**: extract elements from a JSON-formatted string.
- **library**: provides access to GridLAB-D library manager.
- **lock**: provides locking of the current executable to ensure concurrency.
- **manual**: access documentation for HiPAS GridLAB-D.
- **matrix**: provides matrix manipulation and operations using Numpy.
- **openfido**: provides access to OpenFIDO servers.
- **pandas**: provides table manipulation and operations using Pandas.
- **plot**: plots HiPAS GridLAB-D output CSV files to PNG files.
- **python**: provides direct access to HiPAS GridLAB-D's python subsystem.

- **require**: verifies and installs python packages.
- **requirements**: obtains a list of all HiPAS GridLAB-D python requirements.
- **template**: provides access to the HiPAS GridLAB-D template manager.
- **timezone**: resolves timezone specifications.
- **trace**: provides traceback information in case of fatal exceptions.
- **validate**: validates HiPAS GridLAB-D.
- **version**: controls the version of HiPAS GridLAB-D being used.
- **weather**: provides access to the weather data manager.

In addition, a number of tools have been added that can also be run from the command line in a similar manner. These include the following:

- **create_filter**: generates GLM filters from a transfer function specifications.
- **create_player**: generates GLM player objects from input CSV data.
- **create_poles**: generates GLM pole objects for a network model.
- **eia_recs**: downloads RECS data from EIA.
- **find_location**: obtains the latitude and longitude of a location by airport code.
- **fire_danger**: downloads USGS fire danger day-ahead forecast maps for a given date.
- **fire_report**: obtain the fire reports for a state, year, and utility.
- **fit_filter**: generates a GLM filter based on CSV input and output data associated with a putative transfer function.
- **insights**: generates GridLAB-D usage data from the AWS servers.
- **market_data**: obtain wholesale electricity market price and quantity data
- **market_model**: generates a wholesale electricity market model using a transfer function from weather and load data.
- **mdb_info**: outputs class, table, and exports of specific tables in MDB files.

- **metar2glm**: obtains realtime weather data from the FAA.
- **meteostat_weather**: downloads historical weather data from Meteostat.
- **noaa_forecast**: obtains real-time weather forecasts from NOAA.
- **nsrdb_weather**: downloads historical weather data from NREL's NSRDB archive.
- **ucar_weather**: UCAR weather data package for METAR

Fast Installation

HiPAS GridLAB-D utilizes a package-style installation structure, enabling the creation of pre-built images that can run on supported operating systems with minimal installation required. Images are currently hosted on AWS S3.

- **Windows:** Support is available through Microsoft WSL for Debian 9, 10, 11, and Ubuntu 20.04.
- **Linux:** Support is available for Debian 9 on x86_64 systems,and also for Debian 10, 11, and Ubuntu 20.04 on both x86_64 and ARM64 operating systems.
- **MacOS:** Support will be available for Darwin 19, 20, and 21 on Intel chips and Darwin 21 for the ARM64 architecture.

In addition, the following tools have been created to support building and installing HiPAS GridLAB-D. These tools have been included in the main HiPAS GridLAB-D repository.

- **build-image.sh:** Saves the package directory and compiled binaries into a tar file for upload to aws.
- **web-install.sh:** Automatically downloads the correct image for your operating system, if available, and installs any system dependencies that cannot be included within the package.

Cloud Operations

HiPas GridLAB-D utilizes a suite of infrastructure tools for ease of deployment and user access.

- **GitHub:** The main source code for GridLAB-D is stored on github and is accessible for users looking to build the application themselves or report issues,

and developers looking to contribute to the project. GitHub Actions is utilized to maintain CI/CD for the project.

- **Amazon AWS:** Supports a variety of online resources for GridLAB-D and the GridLAB-D website. This includes publicly available documentation, fast-install images for supported operating systems, tutorials and other support resources.
- **DockerHub:** Provides reliable and consistent containers to run GridLAB-D and conduct simulations in a variety of environments.