HiPAS GridLAB-D Developer Training Session 1

Introduction

David P. Chassin

Summer 2022

dchassin@stanford.edu

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

HiPAS Developer Training Program

1. **Introduction**
2. Building GridLAB-D
3. GridLAB-D modeling
4. Testing and validation
5. Modules and classes
6. Core engine and solvers
7. Python libraries and extensions
8. Data and model converters
9. Geodata packages
10. Subcommands
11. Tools and utilities
12. Production and release

History

SLAC

Prototype from PNNL (2003)

- Joined building thermal and grid powerflow solvers

Original version developed at PNNL (2007)

- Funding initially from DOE Office of Electricity
- Goal to study emerging smart grid technology

Some early studies/results

- Taxonomy feeders
- Conservation voltage reduction
- Transactive energy system modeling

Recent additions from PNNL (not included in HiPAS)

- Transient dynamic solvers
- HELICS co-simulation

Why HiPAS?

SLAC

CEC Objectives

- Improve performance of GridLAB-D
- Increase usability for non-experts
- Improve developer support tools/platforms/delivery
- Increase use of ML, AI, and big data
- Address emerging California use-cases
- Support a new user-interface (GLOW)
- Support a new data exchange tool (OpenFIDO)
- Commercially supported version for major CA utilities
- Grow work-force that can support GridLAB-D

HiPAS GridLAB-D Use-Cases

SLAC

Original use-cases

- Integration Capacity Analysis
- Locational Net Benefit Analysis

Final use-cases

- Hosting Capacity Analysis
- Resilience
- Electrification
- Tariff Design

Agent-based simulations

SLAC

What does agent-based mean?

- Each object modeled independently
- Interactions between individual instances are explicit
- Environment *emerges* from interactions of agents

Why use agent-based models?

- Alternative to simulations that implement "the" solution
- Useful when explicit solutions cannot be derived

Notional Example: Lotka-Volterra system

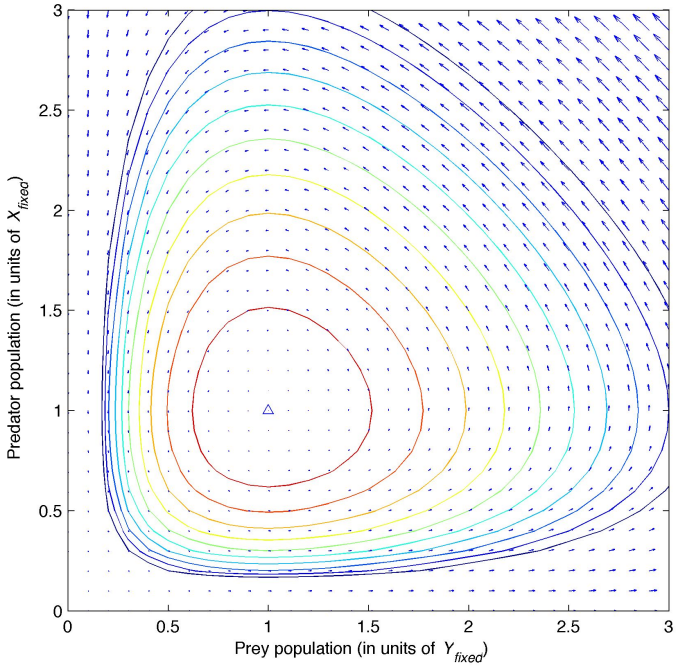
SLAC

Predator-prey population dynamics

$$\dot{x} = x(a - by)$$

$$\dot{y} = y(cx - d)$$

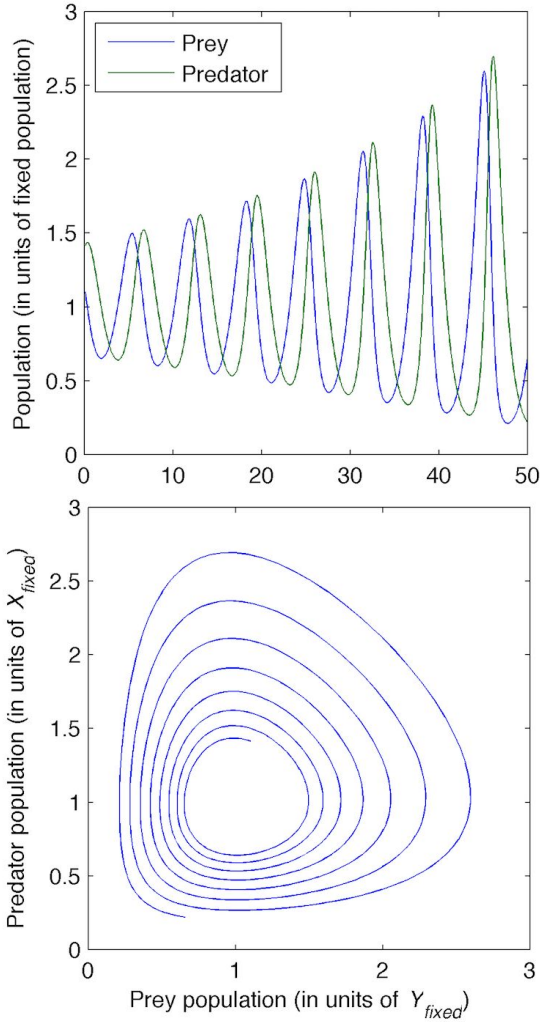
$$y = \frac{a}{b}, \qquad x = \frac{d}{c}$$



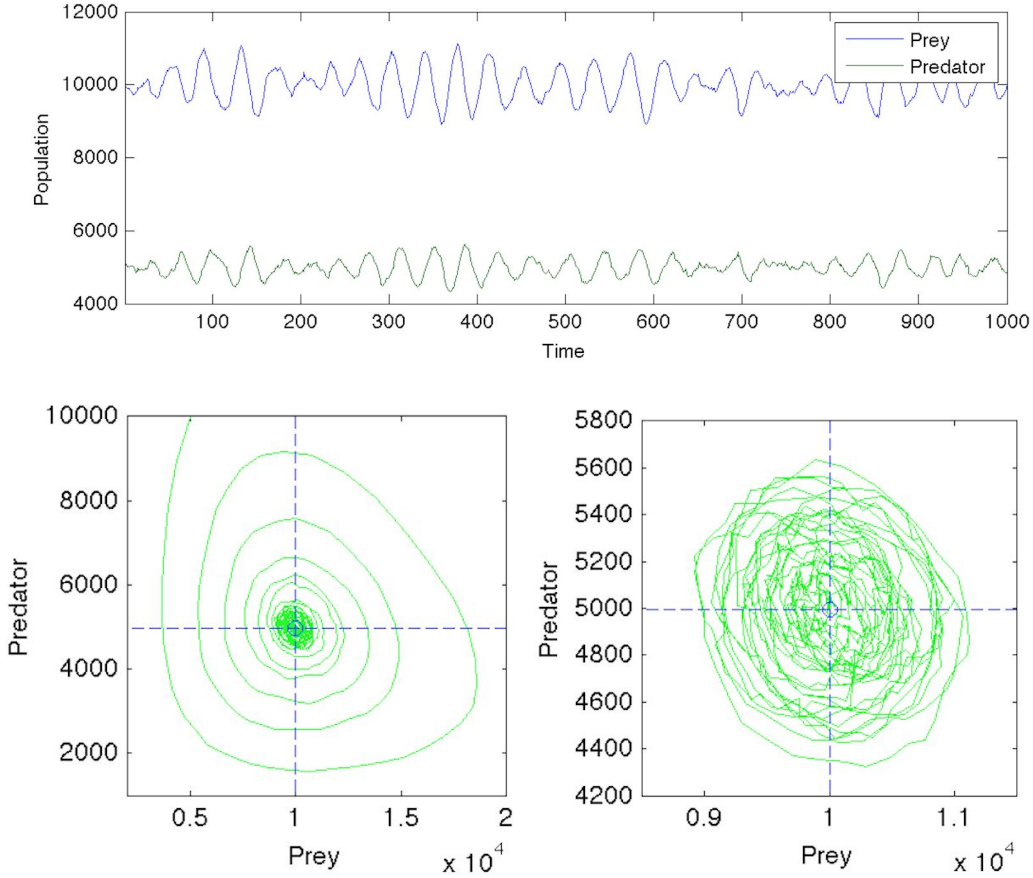
Numerical simulations

SLAC

Naive Finite Difference



Agent-based

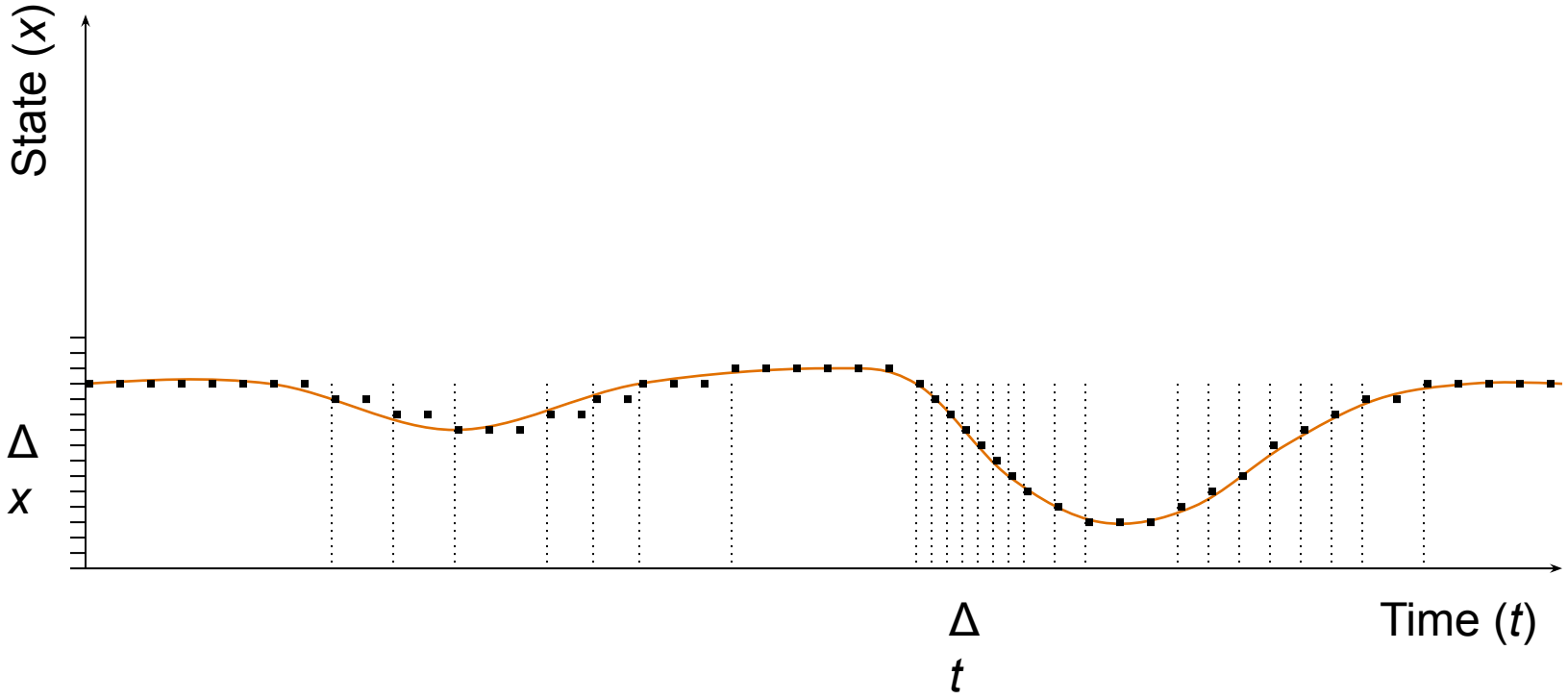


Source: Chassin et al, "*GridLAB-D: An agent-based simulation framework for smart grids,*" Journal of Applied Mathematics, 2014-06-23 (2014)

GridLAB-D simulation method

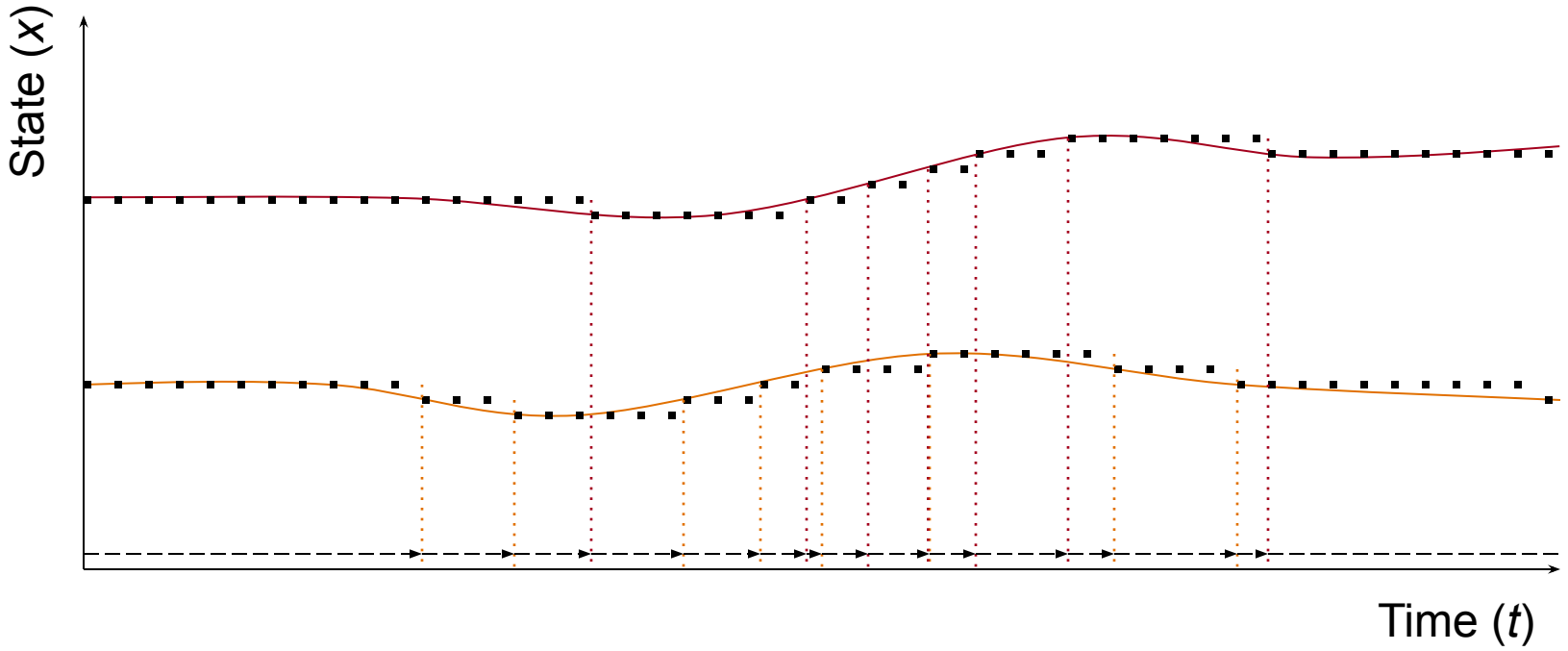
SLAC

Time-series** of **steady states** at **resolution Δx
*with **variable time-step** of **resolution Δt***



Determining next state change

Solution of multiple overlapping quasi-steady time series sub-solutions



Consequence of methodology

All agents must be able to solve both

$x = F(t)$

and

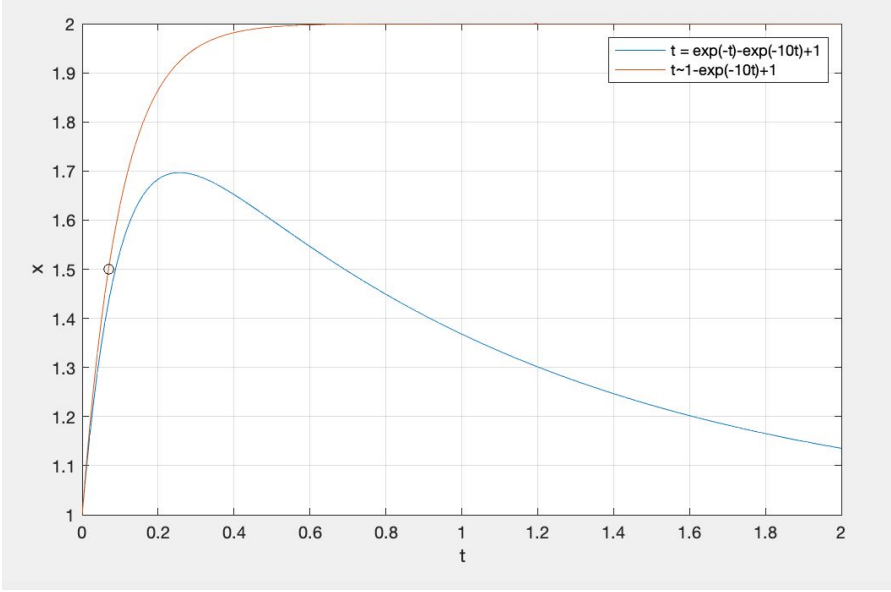
$t = F^{-1}(x)$

This can be challenging to implement, e.g.,

$x(t) = \exp(-at) - \exp(-bt) + c$

Approximations do work, e.g.,

$t(x) > -\log(x-c-1)/b$ for $a < b$



Caveats

Iterations are often required to find next state

- Non-causal behavior is allowed

No guarantee of convergence

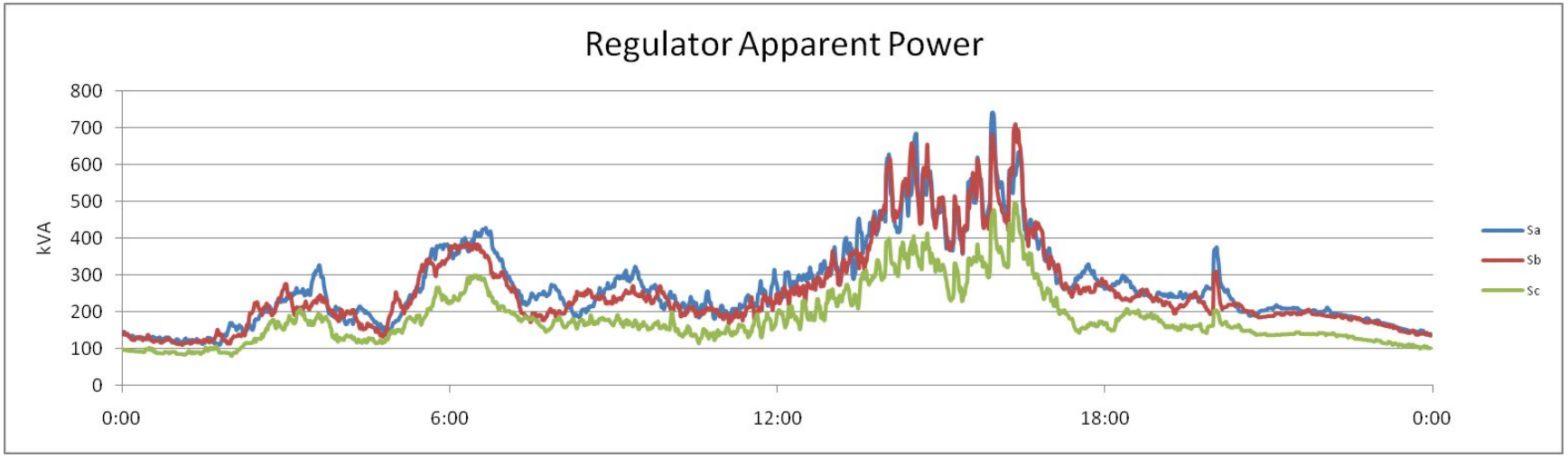
- Multiple simultaneous interacting models
- Incomplete information exchange
- Presence of integral/discrete state variables

Numerous layers of interaction

- Makes diagnostics difficult
- Can lead to unintended consequences

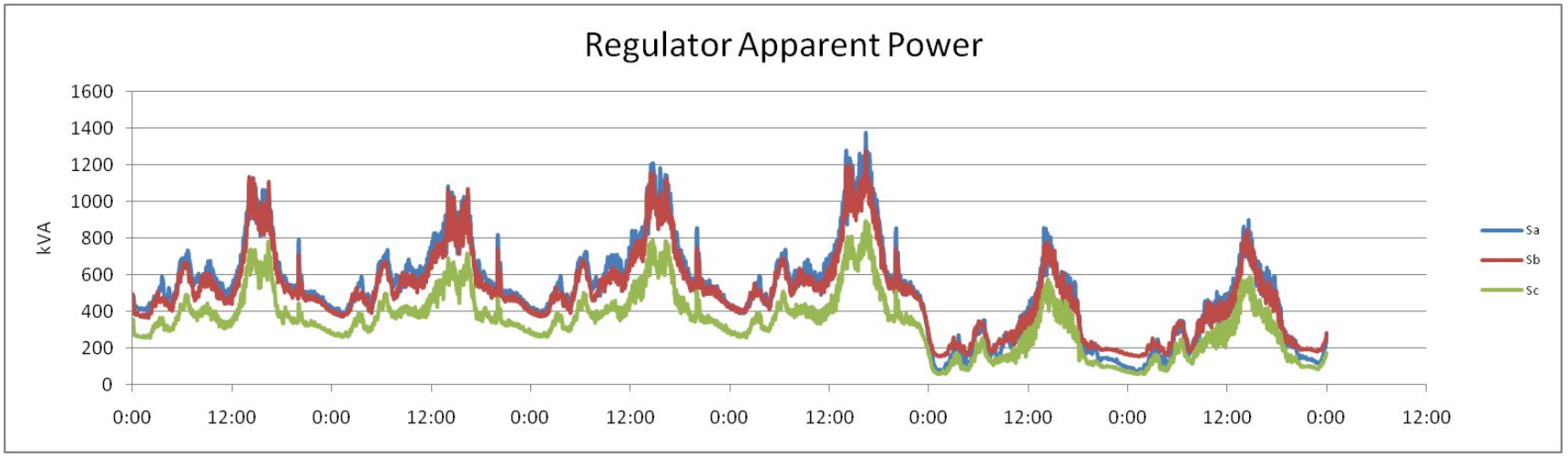
Basic concepts: Daily Cycles

- Diurnal changes (e.g., solar, temperature, humidity)
- Human behavior (e.g., home, work, sleep)
- Scheduled behavior (e.g., thermostat set-point changes)



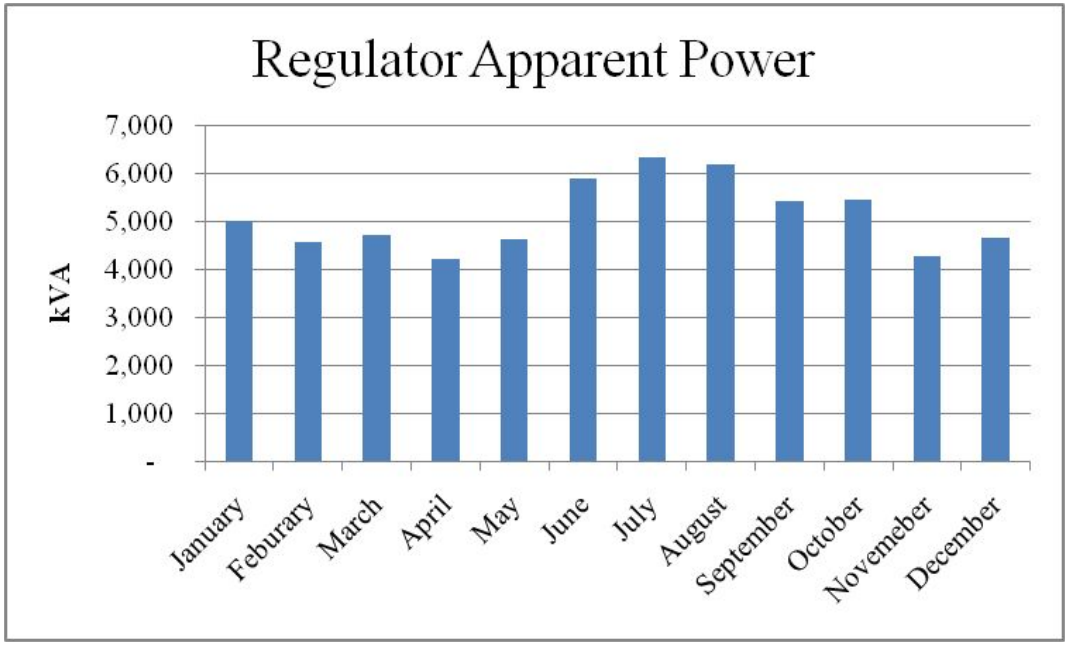
Weekly Cycles

- Weekend/weekday schedules
- Weather patterns



Seasonal Cycles

- Climate (seasonal weather patterns)
- Daylight savings time (minor effect)
- Tariffs (affects costs)
- Loading/congestion (affects controls)



Frequently needed information

SLAC

Voltage profiles

- Regulated voltage
- Voltage drop

Loads

- Feeder total load, power factor
- Sum of residential loads
- Sum of commercial loads

Weather

- Temperatures, winds, solar
- Precipitation, clouds, snowpack

Losses

- Overhead lines
- Underground lines
- Triplex lines
- Transformers

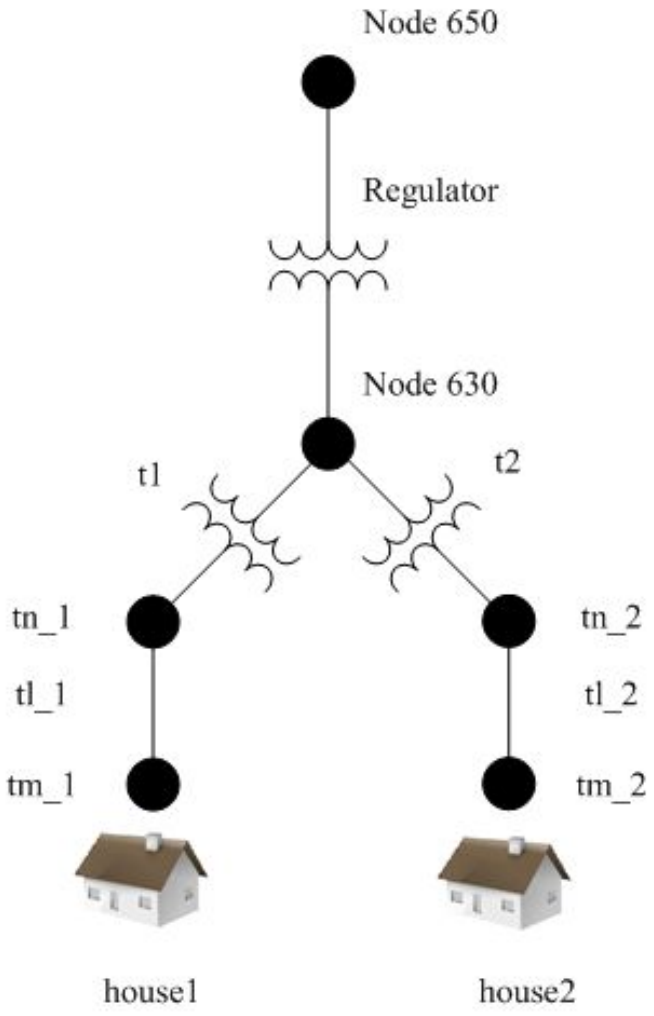
Statistics

- Energy prices / costs
- HVAC unit states
- Appliance states

Examples

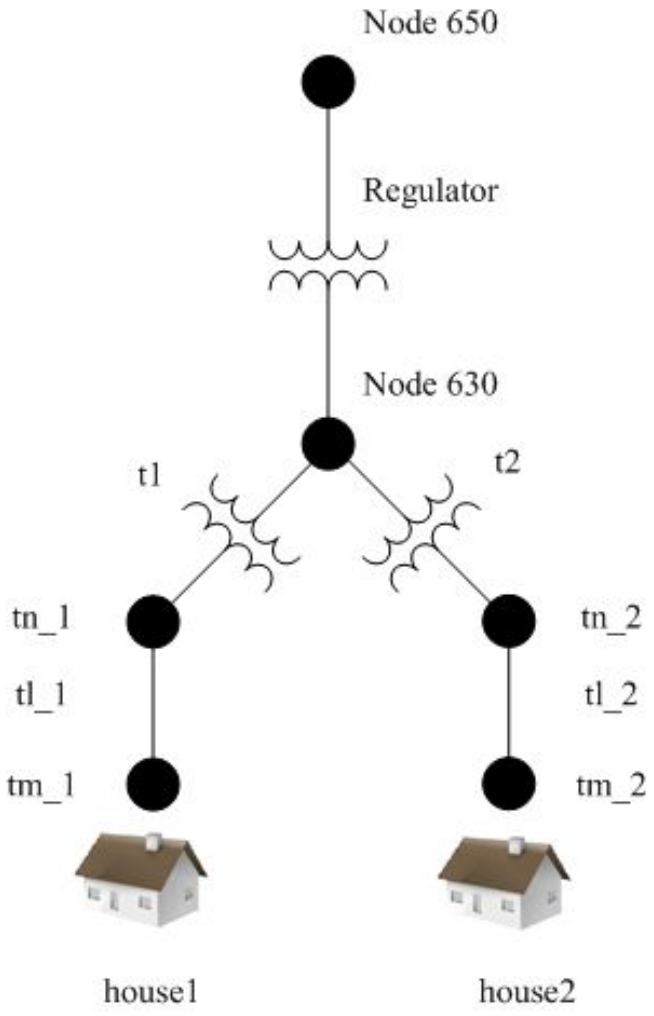
See www.gridlabd.org for studies referenced here

A Simple System



- House 1 is on phase A
  House 2 is on phase B.
- Compare behavioral differences between 2 houses.
- Even small systems can generate an enormous amount of output data.

A Simple System



Typical model elements

- Secondary distribution
- Residential components

Simulation for 1 year

- Takes several minutes to run
- Requires over 50 recorders
- 525,600 power flow solutions
- Each object has a solution for each time step

When simulation run too long

SLAC

Large distribution feeder with 2,000 residential homes

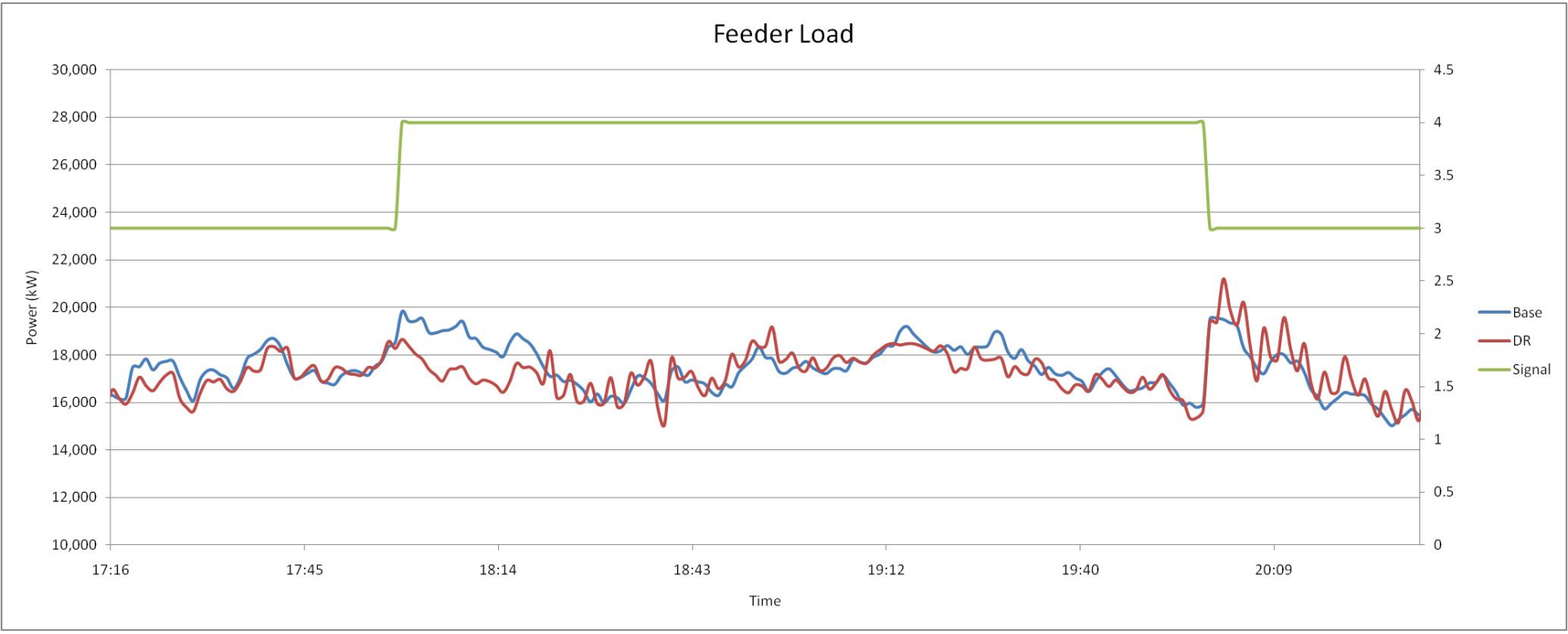
Simulations can be broken into multiple parts

- By week (52 simulations = 1 year)
- By month (12 simulations = 1 year)

Each part must be the exact same file with continuous schedules

- When a simulation is broken into multiple parts the thermal history of the loads is lost at these points.

Demand Response



Prototypical Feeders

SLAC

Feeder	Base kV	Peak MVA	Description
R1-12.47-1	12.5	5.4	Moderate suburban and rural
R1-12.47-2	12.47	4.3	Moderate suburban and light rural
R1-12.47-3	12.47	2.4	Small urban center
R1-12.47-4	12.47	1.8	Heavy suburban
R1-25.00-1	24.9	4.9	Light rural
R2-12.47-1	12.47	2.3	Light urban
R2-12.47-2	12.47	6.7	Moderate suburban
R2-12.47-3	12.47	6.7	Light suburban
R2-25.00-1	24.9	4.8	Moderate urban
R2-35.00-1	34.5	21.3	Light rural
R3-12.47-1	12.47	6.9	Heavy urban
R3-12.47-2	12.47	11.6	Moderate urban
R3-12.47-3	12.47	4.0	Heavy suburban
R4-12.47-1	13.8	9.4	Heavy urban with rural spur
R4-12.47-2	12.5	6.7	Light suburban and moderate urban
R4-25.00-1	24.9	2.1	Light rural
R5-12.47-1	13.8	1.0	Heavy suburban and moderate urban
R5-12.47-2	12.47	10.8	Moderate suburban and heavy urban
R5-12.47-3	13.8	4.2	Moderate rural
R5-12.47-4	12.47	4.8	Moderate suburban and urban
R5-12.47-5	12.47	6.2	Moderate suburban and light urban
R5-25.00-1	22.9	8.5	Heavy suburban and moderate urban
R5-35.00-1	34.5	9.3	Moderate suburban and light urban
GC-12.47-1	12.47	12.1	Single large commercial or industrial

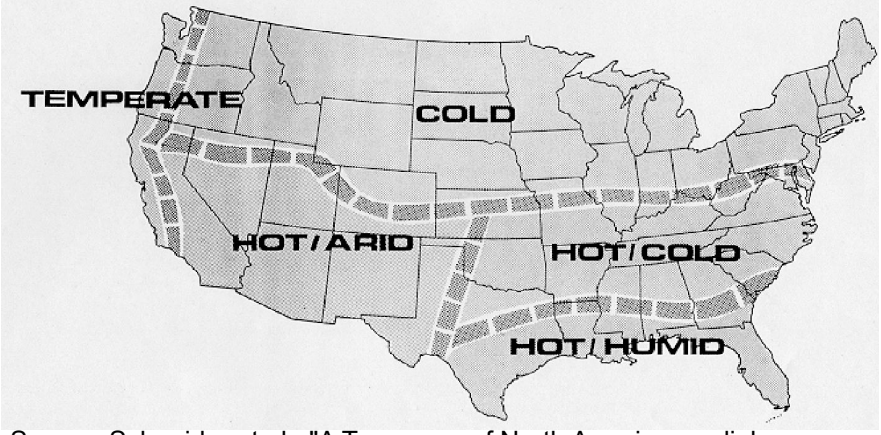
5 climate regions

3 general voltage levels

- 12.47kV
- 25.00kV
- 35.00kV

23 regional feeders

- 1 commercial feeder



Source: Schneider et al., "A Taxonomy of North American radial distribution feeders," IEEE PES GM (2009).

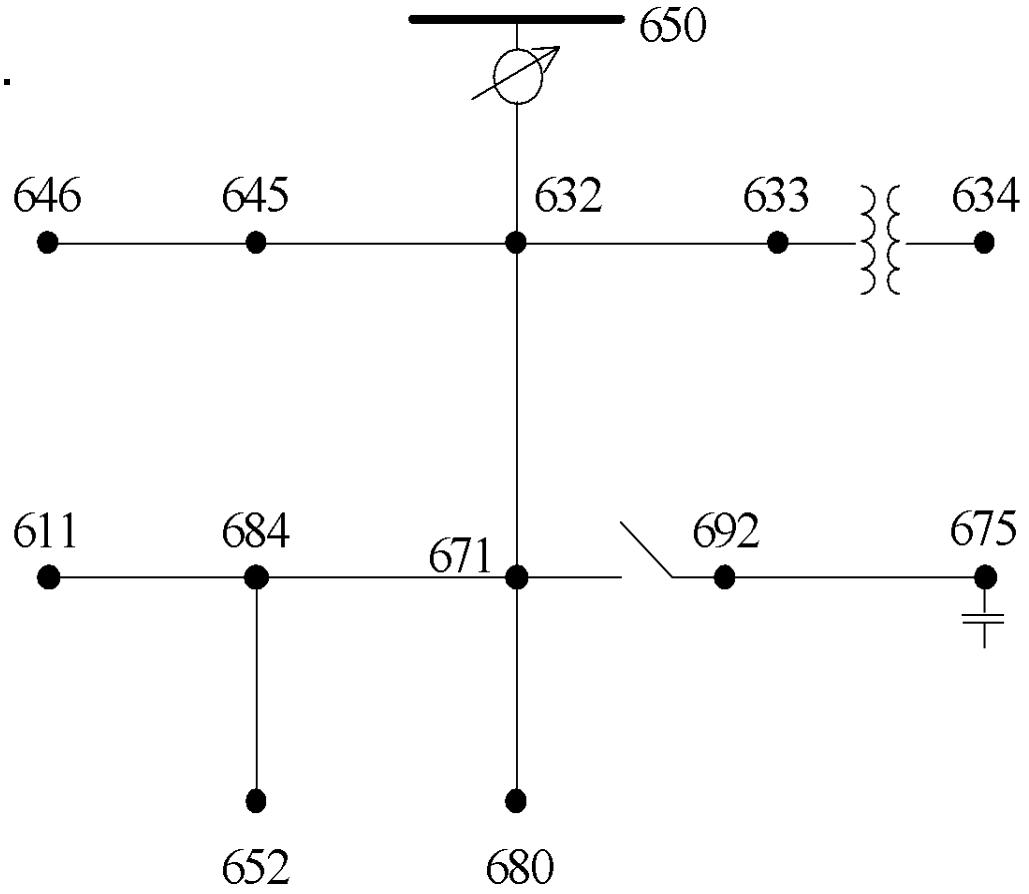
Integrated Volt/VAR Control (IVVC)

SLAC

Openly available Volt-VAR control system

- Implemented in GridLAB-D.

V. Borozan, M. Baran, and D. Novosel, “Integrated Volt/VAR Control in Distribution Systems”, *IEEE PES Winter Meeting*, 2001.



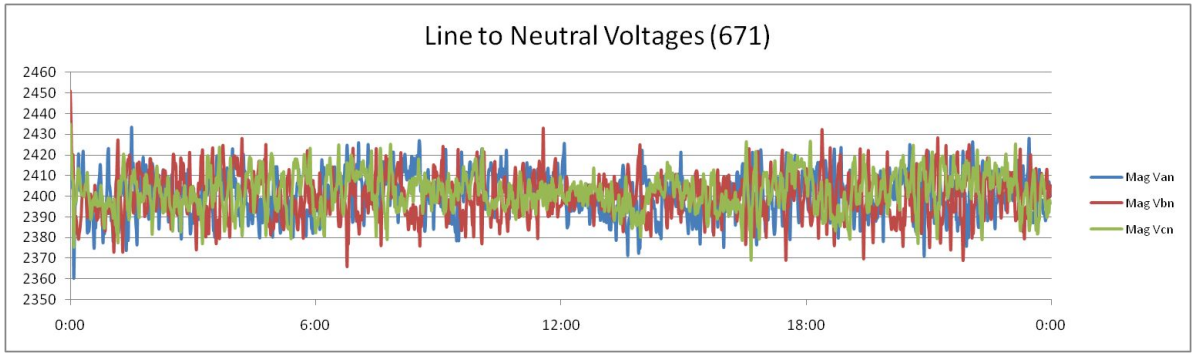
Composed of 2 coordinated goals.

- Voltage reduction
- VAR control

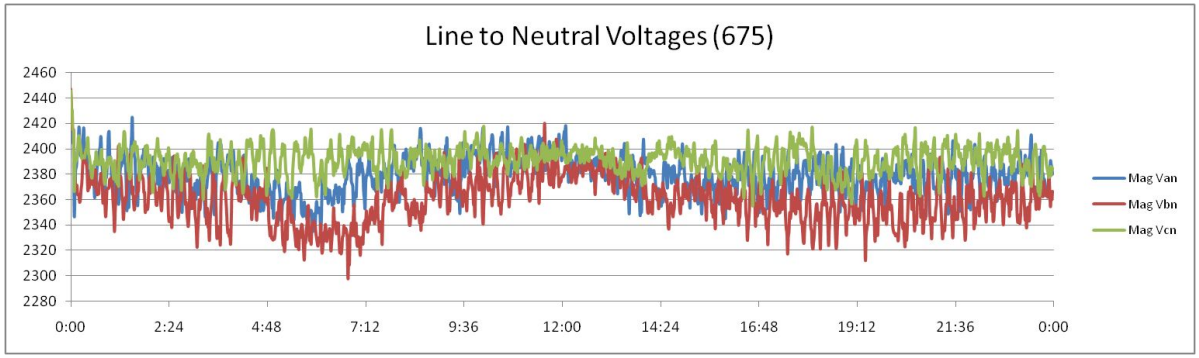
Source: Schneider et al., "A Method for Evaluating Volt-Var Optimization Field Demonstrations", IEEE TSG 5:4 (2014)

IVVC Base Case: Local Voltage Control

Voltage at node:671 is controlled by regulator to 2,400V:



Voltage at node:675 is maintained by capacitors:



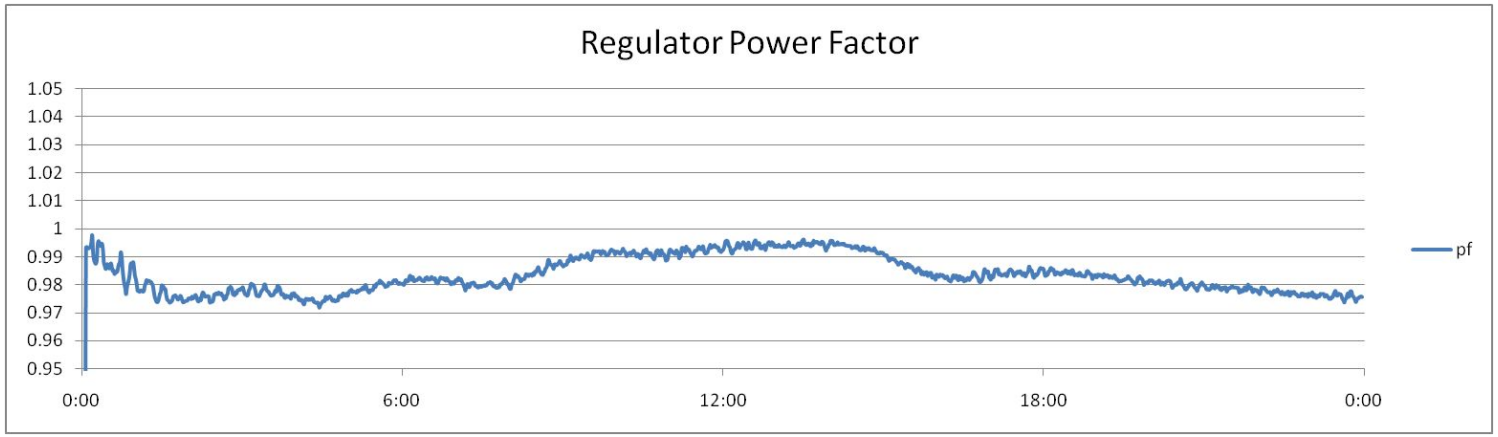
IVVC Base Case: Local Voltage Control

SLAC

Voltage is still operated at a higher level → energy savings opportunity

P (kWh-hr)	38,734
Q(kVAR-hr)	7,185
S(KVA-hr)	39,440

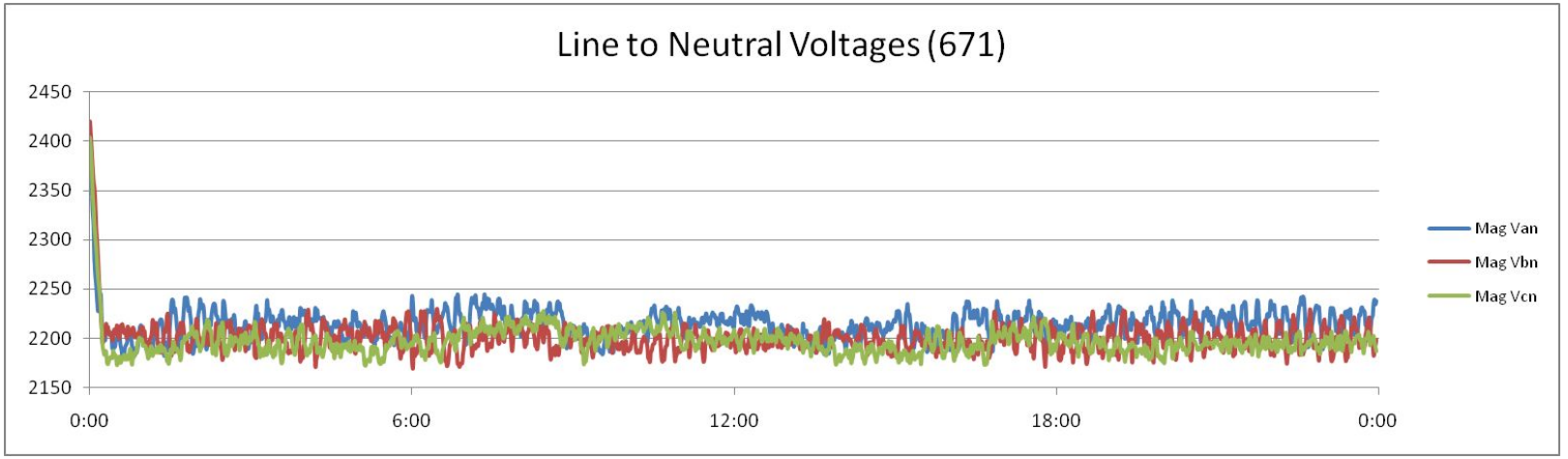
Power factor is not maintained:



IVVC Study: Coordinated Volt/VAR Control

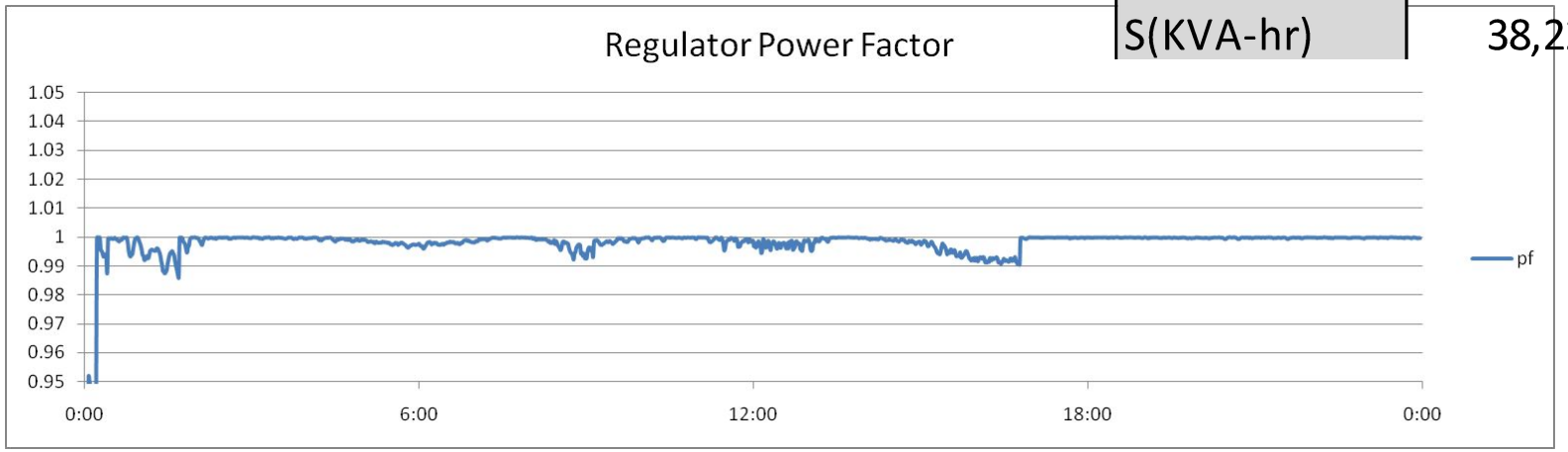
SLAC

Voltage is regulated at remote nodes 652 and 680, to a values of 2,200V:

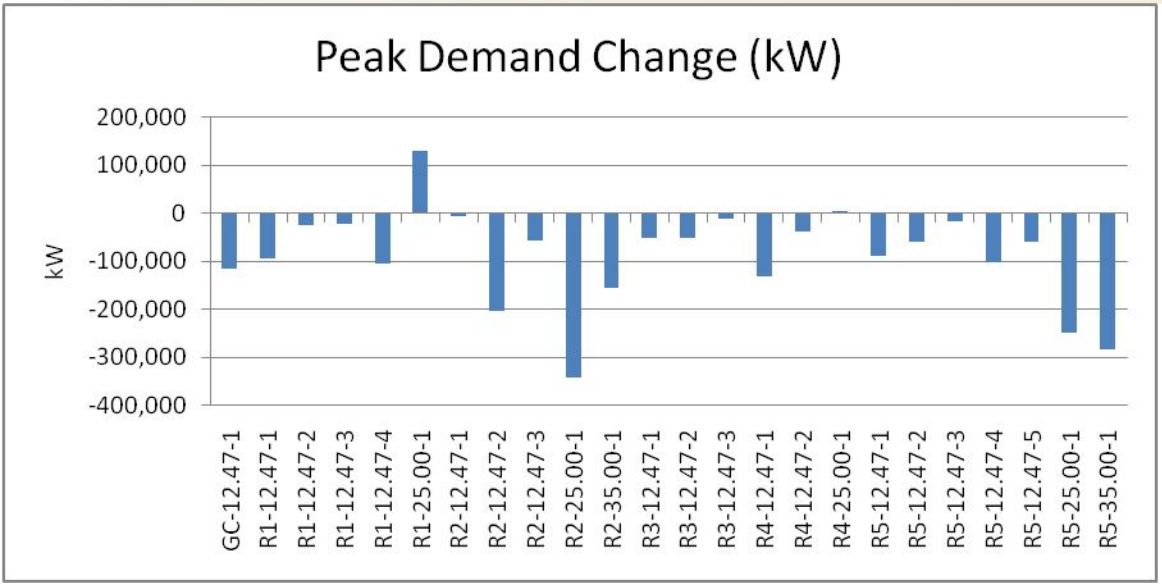


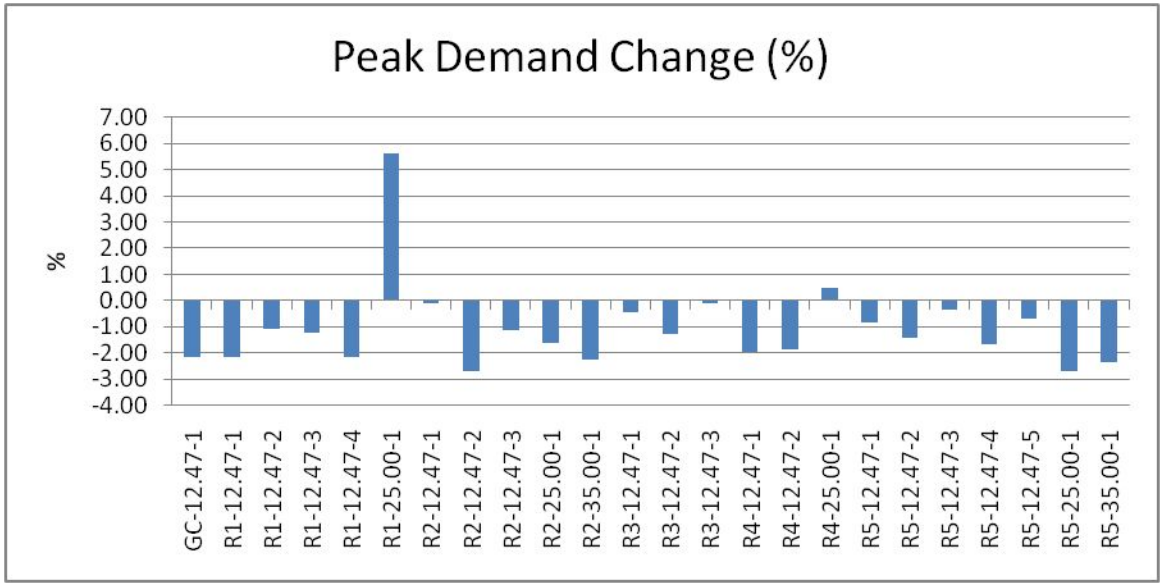
Energy consumption is reduced:

P (kWh-hr)	38,088
Q(kVAR-hr)	778
S(KVA-hr)	38,221

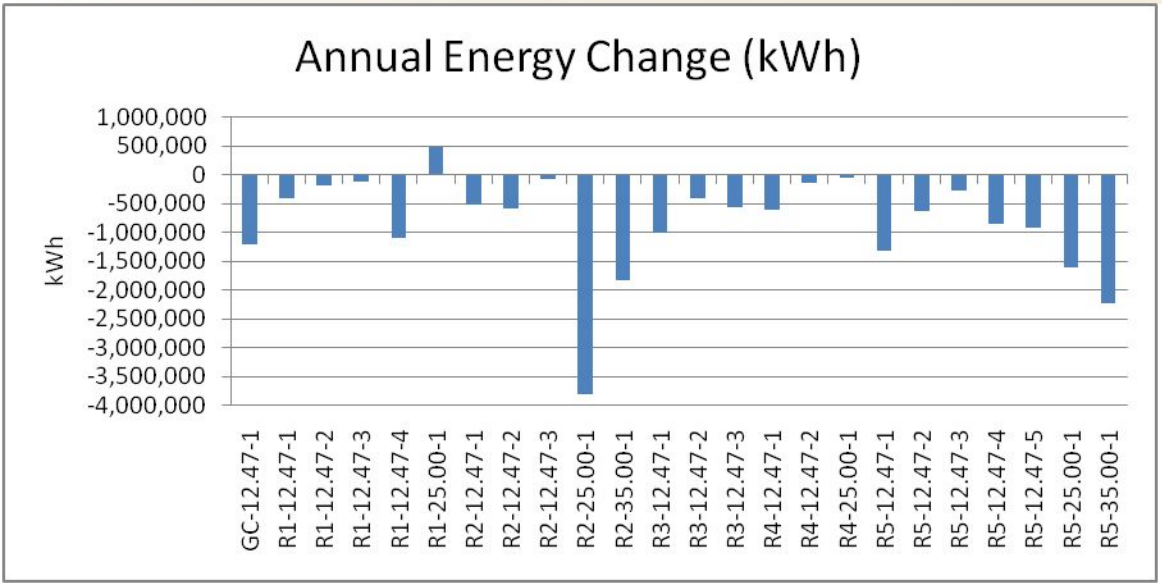


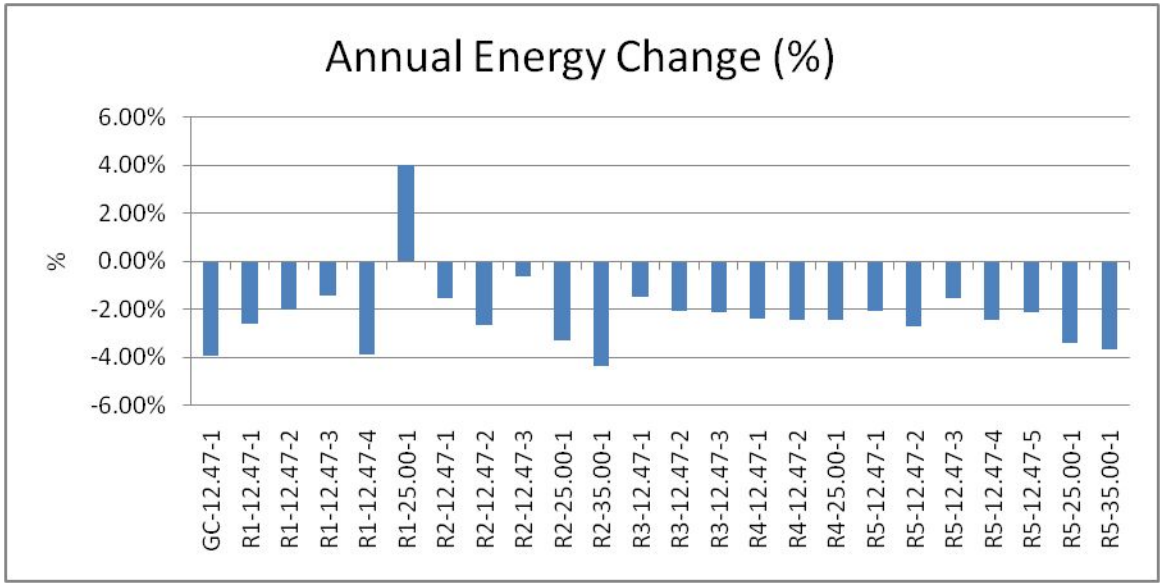
IVVC: Peak Demand Change





IVVC: Annual Energy Change





Break

Return in 5 minutes

Modeling Language

GLM Files

SLAC

GLM files describe simulations

- Input files (**G**rid**L**AB-D **M**odel)
- Classes, objects, players, recorders and clocks
- HiPAS includes automatic converters for many formats

Strengths

- Relatively easy to read and edit
- Rich parametric syntax for managing multiple models
- Single object definitions can become entire populations
- Object properties can have statistical distributions

Weaknesses

- Not directly compatible with any major software tools

What is in a GLM File?

SLAC

Declares classes and objects used in a model

- Standard classes are implemented in modules
- All classes can be extended at runtime
- New classes can be defined at runtime

Defines overall model structure

- Establishes initial conditions
- Defines certain boundary conditions and internal connections
- Sets starting time and stopping time

Includes input and output agents for simulation

- Players, recorders, collectors are objects too

What is a module?

SLAC

Collects classes that are related

- Delivers all model functionality related to a particular domain
- Loaded on demand
- Groups common parameters and domain solvers
- Often includes module globals to control solver options

May be proprietary or open-source

- All modules are currently open-source and included
- Vendor-based distributions possible (none so far)

Modules Available

System modeling

- climate
- commercial
- connection
- generators
- industrial
- market
- powerflow
- reliability
- resilience
- revenue
- tariff
- transactive

Simulation utilities

- assert
- influxdb
- mysql
- optimize
- tape

Describing models: classes

SLAC

Classes define similar objects

- Properties are the same for all objects
- Behavior can be set parametrically
- Exposed methods (if any) are shared

Static classes are precompiled into modules

- Hard to build but can be very rich, detailed, and efficient

Runtime classes are user-defined

- Easier to build but harder to make detailed and efficient
- Best implemented in Python

Describing systems: objects

SLAC

Objects are instances of classes

- Single instance or population of instances

Object properties

- Explicit property values or distribution of values

Parent-child relation

- Dictates primary solver dependency relationship
- Child dependent on parent before parent dependent on child
- Child expected to get/put information from/to parent

Example Object Declaration

SLAC

Define a house with default values

```
module residential; // module declaration
object house { // instantiation of an object of class house
  name "MyHouse"; // name the new object
}
```

Define a house with custom values

```
module residential;
#include "CA-Los_angeles.glm" // load weather
object house {
  name MyHouse;
  floor_area random.triangle(1000,2000) sf; // random values
  heating_setpoint 70.0 degF; // heating setpoint
  cooling_setpoint 76.0 degF; // cooling setpoint
}
```

Controlling time: clocks

SLAC

System clock represents simulation time

- Clock time resolution is 1 sec or greater
- Cannot represent any time before 1970-01-01 00:00:00 UTC
- TS_INIT is *2000-01-01 00:00:00 UTC*
- Objects have private synchronization clocks

Local time based on time zones

- Use POSIX standard (e.g., PST+8PDT)
- Alternative localization (e.g., US/CA/San Francisco)

Summer (daylight savings) time rules are handled

- Summer time rule change years are supported

Clock

Establishes the simulation time range for the model

```
clock {
   timezone "PST+8PDT";
   starttime "2000-01-01 0:00:00 PST";
   stoptime "2001-01-01 0:00:00 PST";
}
```

If clock is omitted TS_INIT to TS_NEVER is used

- TS_NEVER stoptime is steady state

Example of GLM file with clock

SLAC

```
clock {
  timezone PST+8PDT;
  starttime '2000-01-01 0:00:00 PST';
  stoptime '2000-07-01 0:00:00 PST';
}

module residential;

object house {
  floor_area random.normal(1500,300) sf;
  heating_setpoint 70.0 degF;
  cooling_setpoint 76.0 degF;
  thermostat_deadband 1.0 degF;
}
```

Running a simulation

SLAC

To load and run a GLM file

- Command line must point to location of the executable

```
host% gridlabd run_file.glm
```

Output by default goes to current working folder

- Only saves final state of the system

Sending output to another file

```
host% gridlabd file1.glm -o file1.json
```

- Saves the instance this run created
- Be careful not to overwrite your input file

```
host% gridlabd file1.glm -o file1.glm
```

Categories of output messages

SLAC

Info	Informational only, not very important
Verbose	Useful for diagnosing trivial problems
Debug	Useful for diagnosing serious problems
Warning	Not serious, but could affect results
Error	Serious, affects results, and will stop run
Fatal	Halt condition has been reached
Exception	Problem is unexpected and instantly fatal

Controlling output messages

SLAC

Warnings can be disabled

```
host% gridlabd --warn file1.glm
```

Verbose output can be enabled

```
host% gridlabd --verbose file1.glm
```

Debug messages can be enabled

```
host% gridlabd --debug file1.glm
```

Quiet suppresses almost all messages

```
host% gridlabd --quiet file1.glm
```

Controlling global parameters

SLAC

Many parameters have global or module scope

- Custom model parameters also allowed

Parameters are defined as model globals

```
host% gridlabd -D name=value file1.glm
```

Modules can also have their own globals

```
host% gridlabd -D module::name=value file1.glm
```

- (More about this when using GLM macros)

Some useful global variables

SLAC

iteration_limit	Maximum number of iterations allowed before convergence fails (default is 100)
randomseed	Deterministic pseudo-random number seed; 0 means non-deterministic random numbers (default is 0)
minimum_timstep	The minimum timestep allowed during simulation (default is 1 second)
maximum_synctime	The maximum wallclock time a single sync event is allowed (default is 60s)
maximum_runtime	The maximum wallclock time a simulation is allowed to run (default is 0, i.e., none)
savefile	File to save final result in (default is none)

Reading error messages

SLAC

Model Loader messages

file.glm(*line*): *load message*

Runtime compiler messages

file.glm(*line*): *cpp message*

-or-

file.cpp(*line*): *cpp message*

Runtime simulation messages

ERROR [*timestamp*]: *exec message*

Tools/subcommand messages

ERROR [*name*]: *error message*

Help Resources

SLAC

Source code: https://source.gridlabd.us/

Announcements: https://news.gridlabd.us/

Bug reports: https://issues.gridlabd.us/
Code submission: https://pulls.gridlabd.us/

Developer email: gridlabd@gmail.com

- Ok, but not always as quick
- Other users don't benefit from answer

Runtime resources

- **--help**, **--modhelp** *module*[**:***class*]

User documentation

- Latest documentation at http://docs.gridlabd.us/ (available by branch)
- Searchable using command line query: **gridlabd --info *topic*** (requires GitHub token)

Command line: --help

SLAC

```
host% gridlabd --help
Syntax: gridlabd [<options>] file1 [file2 [...]]

Command-line options
--------------------
  --check|-c                Performs module checks before starting ...
  --debug                   Toggles display of debug messages
  --debugger                Enables the debugger
  --dumpall                 Dumps the global variable list
  --mt_profile <n-threads>  Analyses multithreaded performance profile
  --profile                 Toggles performance profiling of core ...
  --quiet|-q                Toggles suppression of all but error ...
  --verbose|-v              Toggles output of verbose messages
  --warn|-w                 Toggles display of warning messages
  --workdir|-W              Sets the working directory

... more ...
```

Command line: --modhelp tape:player

SLAC

```
host% gridlabd --modhelp tape:player
module tape {
    char1024 gnuplot_path;
    int32 flush_interval;
    int32 csv_data_only;
    int32 csv_keep_clean;
    timestamp delta_mode_needed;
}
class player {
    char256 property;
    char1024 file;
    char8 filetype;
    char32 mode;
    int32 loop;
}
```

Command line: --modhelp

SLAC

```
host% gridlabd --modhelp powerflow:line
...
class line {
    parent link;
    class link {
        parent powerflow_object;
        class powerflow_object {
            set {A=1, B=2, C=4, D=256, N=8, S=112, G=128} phases;
            double nominal_voltage[V];
        }
        function interupdate_pwr_object();
        function update_power_pwr_object();
        function check_limits_pwr_object();
        enumeration {OPEN=0, CLOSED=1} status; //
        object from; // from_node - source node
        object to; // to_node - load node
        complex power_in[VA]; // power flow in (w.r.t from node)
        complex power_out[VA]; // power flow out (w.r.t to node)
        ...
    }
    function interupdate_pwr_object();
    function update_power_pwr_object();
    function check_limits_pwr_object();
    object configuration;
    double length[ft];
}
```

Command line: --example

SLAC

Example returns default defaults:

```
host% gridlabd --example residential:house
object house:0 {

    // header properties
    rank 0;
    clock (invalid);
    flags (invalid);
    // residential_enduse properties
    shape "type: unknown";
    load "power_factor: 0.970000; power.r: 0.000000";
    energy +0+0i kVAh;
    power +0+0j kVA;
    peak_demand +0+0i kVA;
    heatgain +0 Btu/h;
    // ...
}
```

Developer Resources

Online Documentation

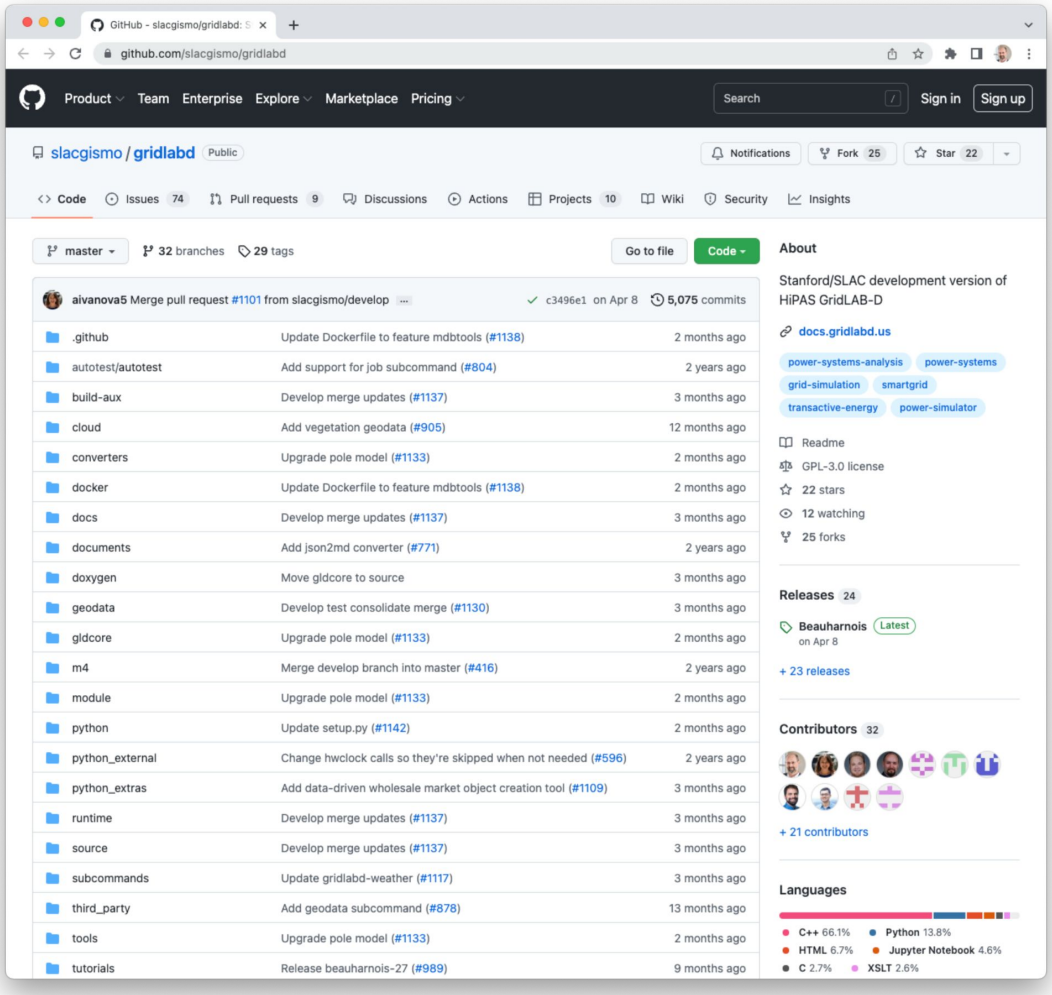
SLAC

https://docs.gridlabd.us/

Source code repository

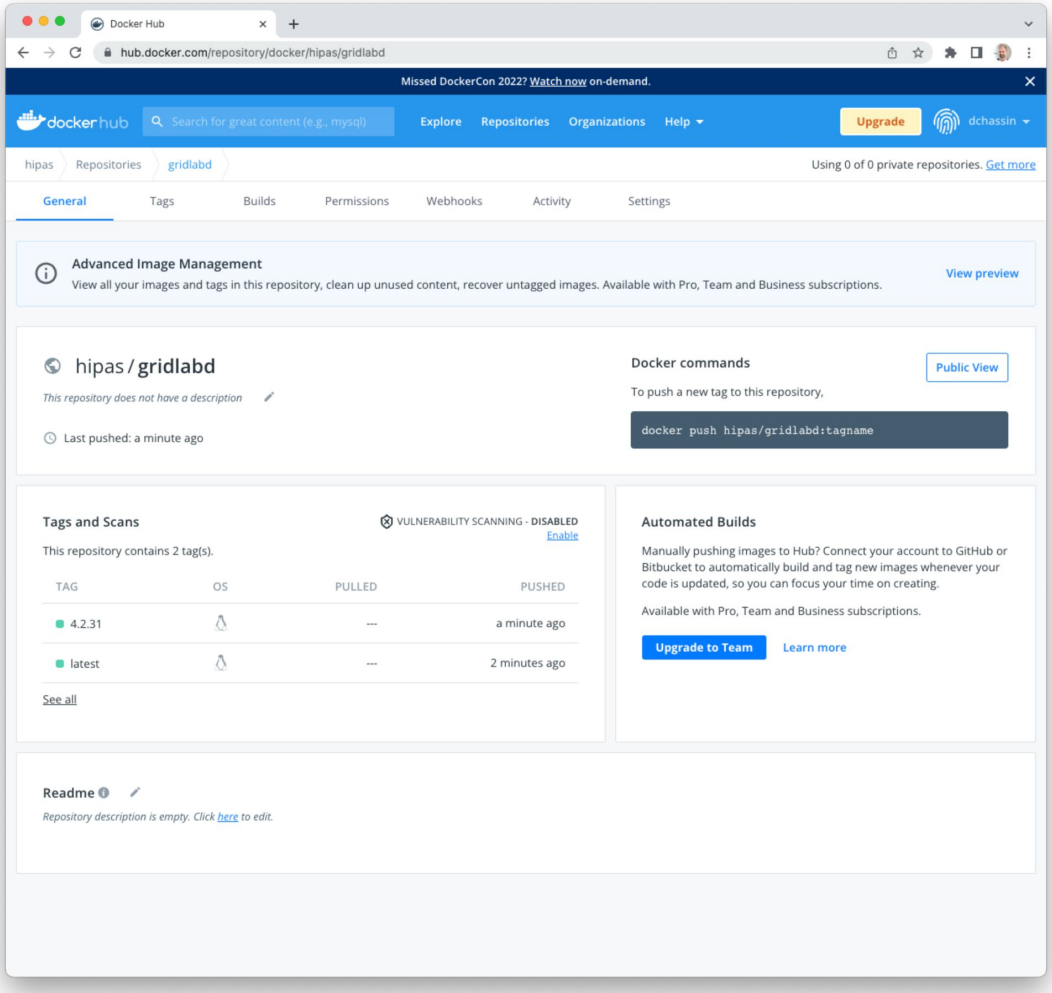
SLAC

https://source.gridlabd.us/



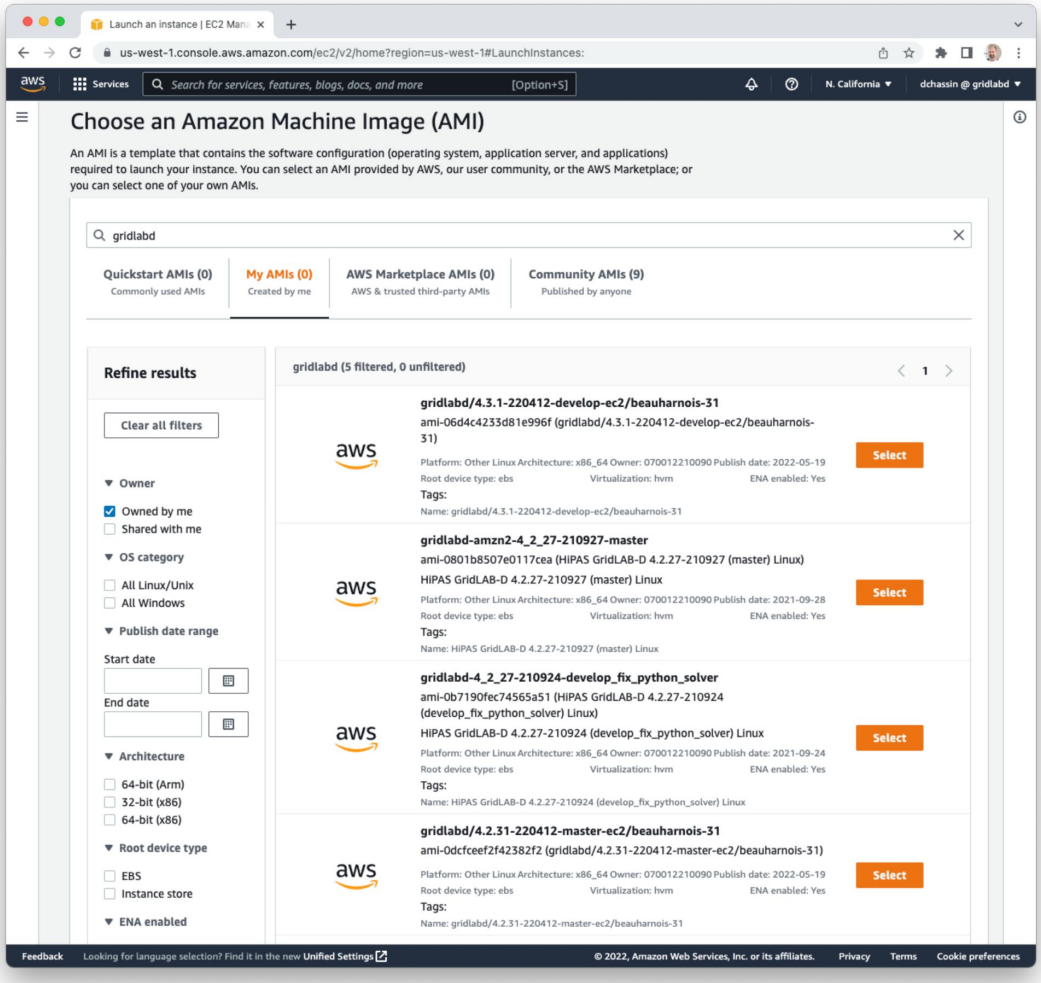
Docker Distributions

Docker hub



Amazon Distributions

Amazon EC2 AMI



Homework Assignment for next session

SLAC

1. **Read the wiki pages on github**
   https://github.com/slacgismo/gridlabd/wiki
2. **Follow the developer install instructions for your platform (e.g., Mac, Windows, Linux)**

Windows users: you will need to run WSL (debian:11)

Questions