GridLAB-D Tutorial – Session 1
Introduction and Fundamentals

David P. Chassin
Summer 2016
dchassin@stanford.edu

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Tutorial objectives

Instruct on GridLAB-D

- Using the simulation
- Adding new models

Addresses the following topics

- Using the GridLAB-D modeling (GLM) language
- Creating models and optimizing simulations
- Administering and configuring GridLAB-D systems
- Creating large-scale GridLAB-D operating environments
- Programming and debugging with GridLAB-D
- Installation and maintenance

What to expect

SLAC

Does not address the following topics:

- How to integrate or link gridlabd with other simulators
- How to change how gridlabd works internally

You will need the following to work problems:

- A system on which to install GridLAB-D (Windows 64, RHEL-6, or Mac OSX Mavericks or Yosemite)
- Install GridLAB-D from SourceForge
- Development tools (e.g., svn, gcc/mingw, gdb, lldb)
- Install Matlab and MySQL
- Course written for Eclipse users, but this is not required
- Optional tools include (e.g., gnuplot) not used in this course.

Tutorial outline

1. Introduction and fundamentals
2. Power systems
3. Advanced loads and weather
4. Markets
5. Generation
6. Reliability analysis
7. Beginning developers – Classes
8. Intermediate developers – Modules
9. Advanced developers – Core

Tutorial structure

SLAC

Instruction

- Objectives
- Theory and concepts
- Methods and applications
- Worked examples

Exercises

- Problems to solve

Questions?

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

What is GridLAB-D?

Background on what GridLAB-D does and how it does it.

Agent-based simulations

SLAC

What does agent-based mean?

- Changing states of each device is modeled independently
- Interactions between individuals are captured
- Environment in which agents evolve *emerges* from interactions

Examples:

- Social sims (e.g., Repast, Swarm)
- Computational economics (e.g., AMES)
- Engineered systems (e.g., TRANSIM)

Not examples:

- Simcity (it's cellular automaton-based simulation)
- Flight simulator (it's a model-based simulation)
- EPRI DSS (implementation of an algebraic solution)
- Any stochastic simulation/model

Notional Example: Lotka-Volterra system

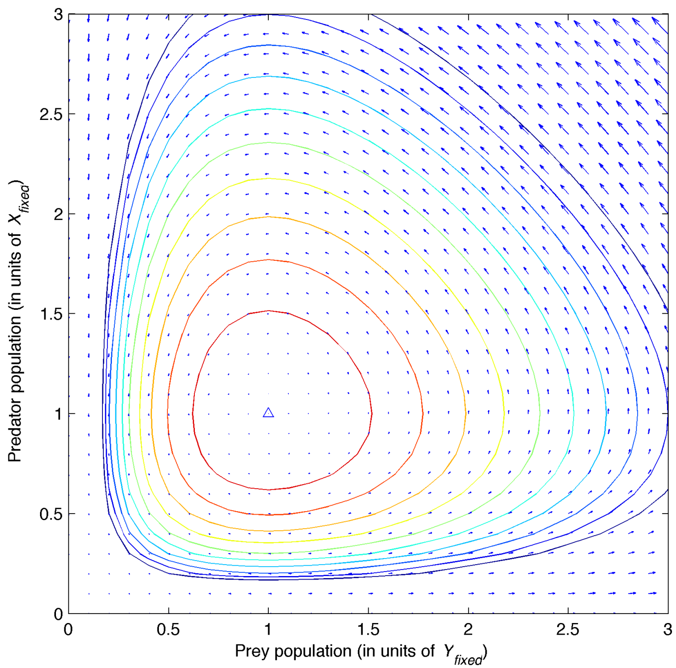
SLAC

Predator-prey population dynamics

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

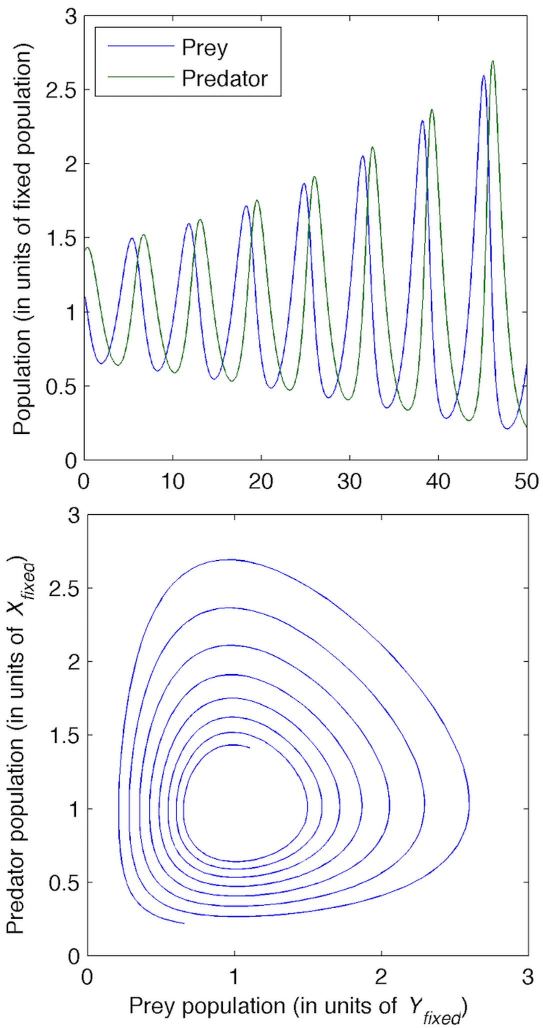
$$\dot{y} = y(cx - d)$$

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

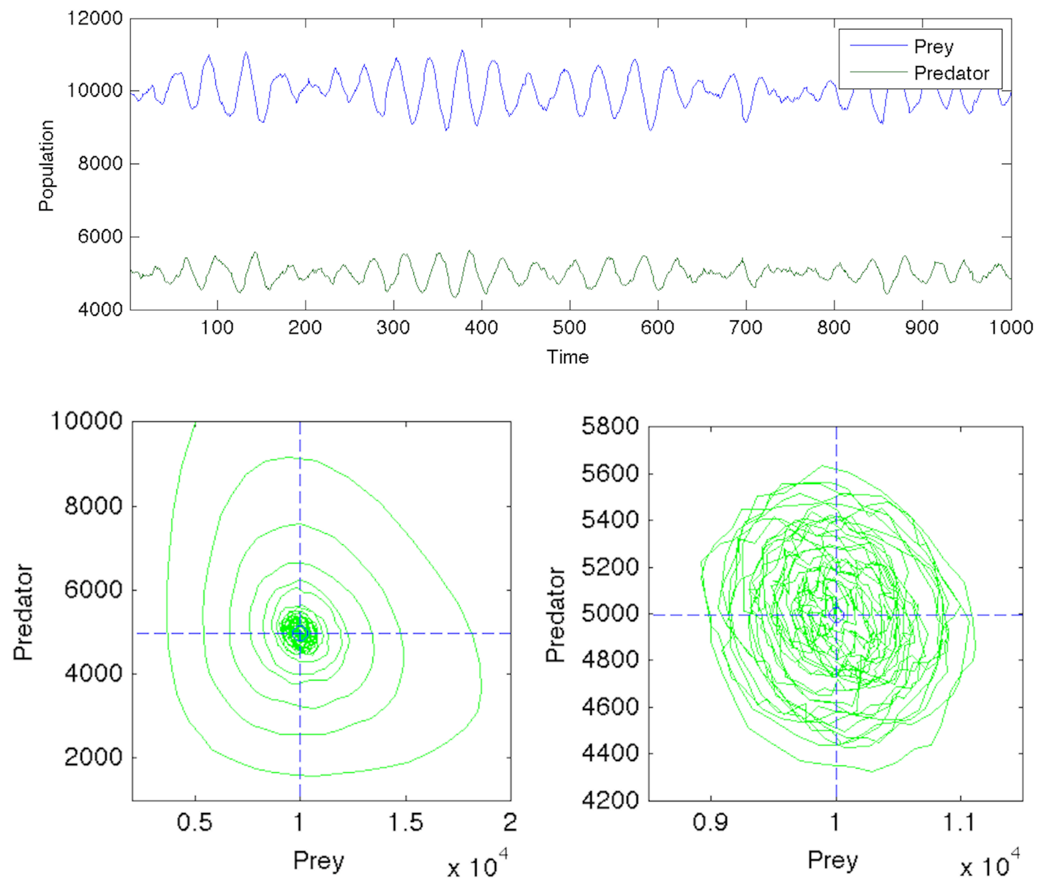


Numerical simulation error

Finite Difference

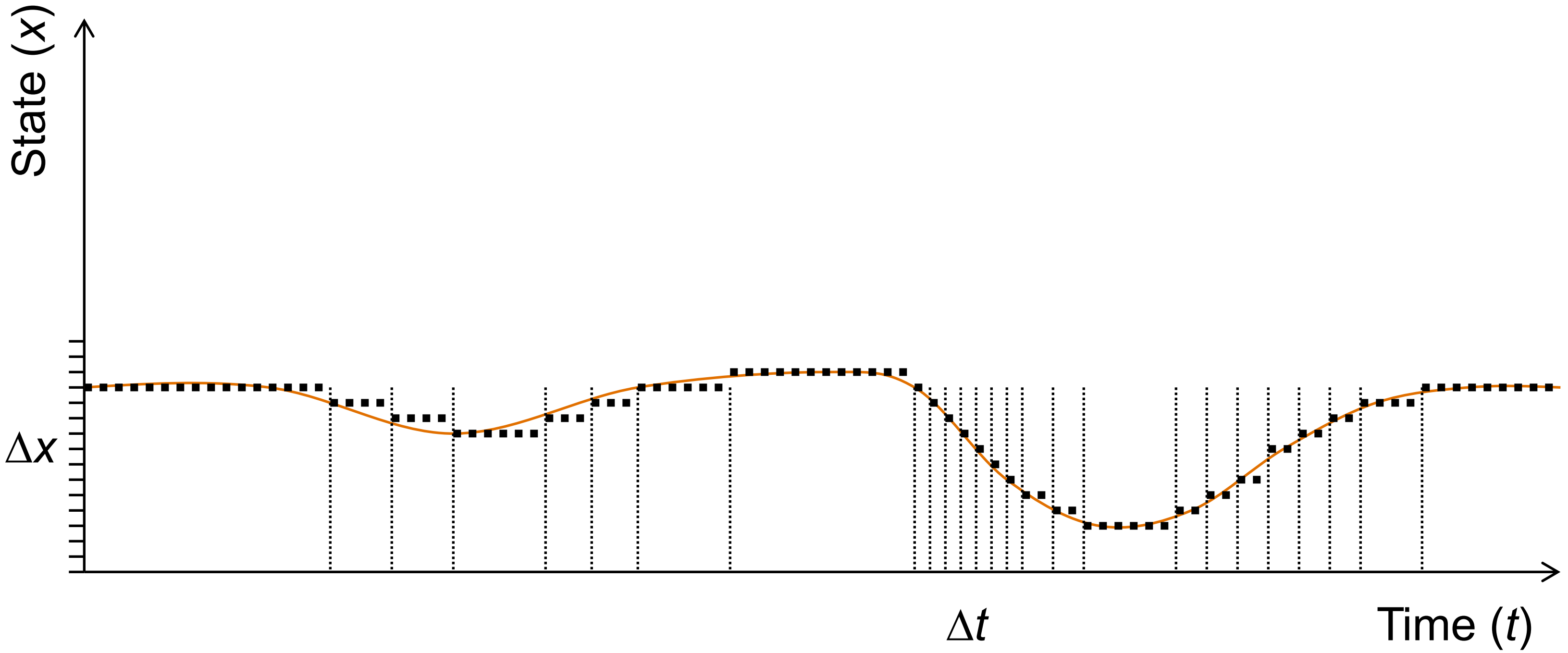


Agent-based



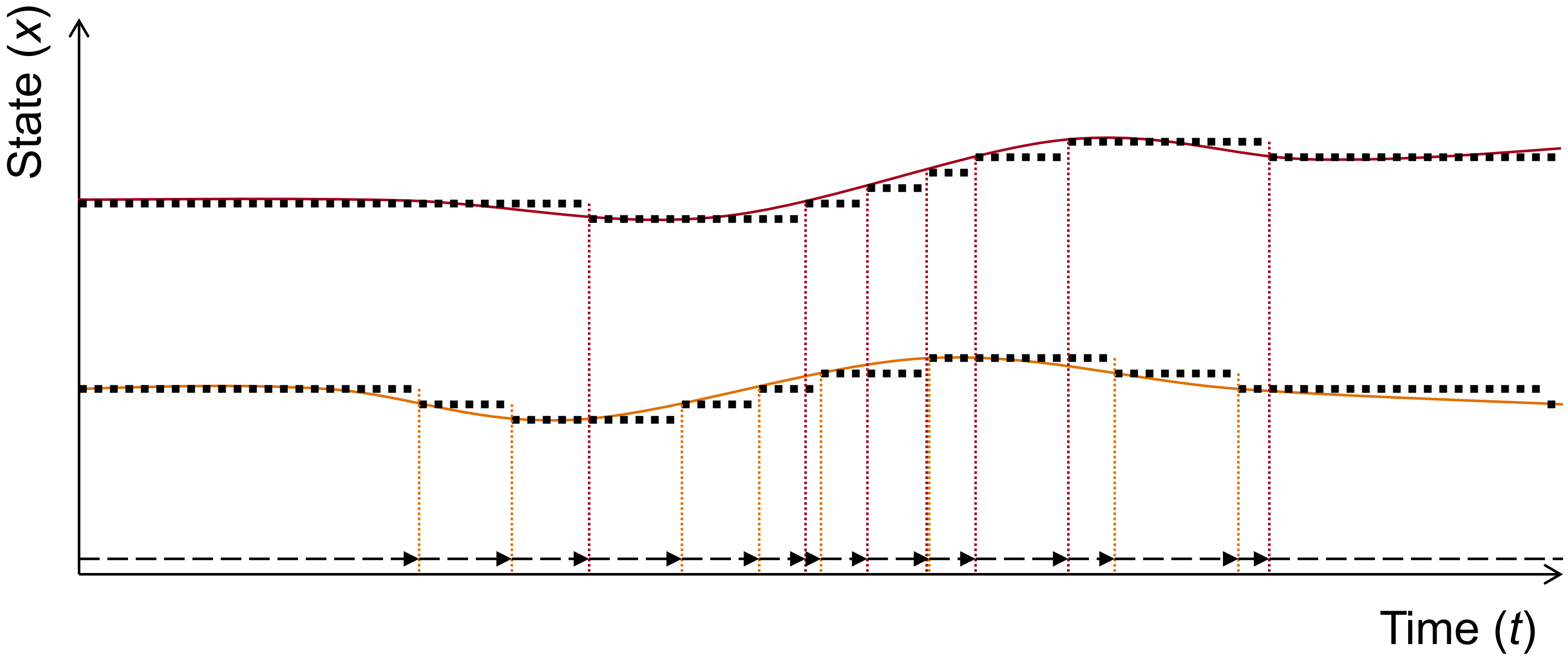
GridLAB-D simulation method

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



Determining next state change

Resolution of multiple overlapping time series



Caveats

Iterations are often required to find next state

- Non-strictly causal behavior is allowed

No guarantee of convergence

- Non-causal behavior is not prohibited
- 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

GLM Files

GLM files describe simulations

- Input files (**G**ridLAB-D **M**odel)
- Classes, objects, players, recorders and clocks

Strengths

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

Weaknesses

- Not directly compatible with any major software tools
- Integration modules/tools have been developed

What is a GLM File?

Declares classes to use

- 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
- Determines starting time and stopping conditions

Constructs inputs and outputs objects

- Players, recorders, collectors are objects too

What is a module?

Collects classes that are related

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

May be proprietary or open-source

- Most modules are downloads from SourceForge
- Vendor-based distributions possible

Modules Available

Climate (incorporates weather data)

Buildings

- Commercial (single-zone office buildings) – *in progress*
- Residential (HVAC and appliance models)

Electrical network

- Powerflow (3 phase unbalanced distribution solver – FBS/NR)
- Reliability (events, metrics)
- Distributed Generators (battery, diesel, PV, wind, inverters) – *generalized models – further development this year*

Markets

- Double and single sided auctions
- Appliance controllers

Input/Output

- Players, recorders, collectors, MySQL, Matlab

Describing models: classes

Classes define similar objects

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

Static classes are precompiled in modules

- Hard to build but very rich and detailed

Runtime classes are user-defined

- Easier to build but harder to make detailed
- Detail is limited by buffer size and ability to interact with other components

Describing systems: objects

Objects are instances of classes

- Single instance or population of instances

Object properties

- Simple value or distribution of values

Parent-child relation

- Primary solver dependency relationship
- Child dependent on parent before parent dependent on child
- Parent expected to aggregate information from children

Example Object Declaration

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.normal(1500,300) sf; // random initial value
  heating_setpoint 70.0 degF; // heating setpoint
  cooling_setpoint 76.0 degF; // cooling setpoint
  thermostat_deadband 1.0 degF; // thermostat hysteresis
}
```

Controlling time: clocks

System clock represents simulation time

- Objects have private synchronization clock
- Clock time resolution is 1 sec or greater
- Cannot represent any time before TS_INIT
- TS_INIT is *Jan 1, 1970 0:00:00 UTC*

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

Establish 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';
}
```

Example of GLM file

```
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;
}
```

Questions?

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Running Simulations

Details on how to run and control GridLAB-D models

Specifying input file

To run a GLM file

- Command line must point to location of the executable

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

Saving output

Output by default goes to current executable location

- Only saves final state of the system

Sending output to another file

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

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

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

Controlling output messages

Warnings can be disabled

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

Verbose output can be enabled

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

Debug messages can be toggled

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

Quiet suppresses almost all messages

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

Controlling global parameters

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

iteration_limit	Maximum number of iterations allowed before convergence fails (default=100)
dumpfile	File to use for model dump if simulation fails (default=gridlabd.xml)
stoptime	Simulation time at which to stop regardless of state (default=never)
kmlfile	Google Earth output file (default=gridlabd.kml)
urlbase	URL base for stylesheets when viewing XML files in a browser (default=http://www.gridlabd.org)
timezone	The default timezone to use if non specified in the model (default=locale)
randomseed	Deterministic pseudo-random number seed; 0 means non-deterministic random numbers (default=0)

Reading error messages

Loader messages

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

Compiler messages

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

-or-

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

Simulator runtime messages

ERROR[*timestamp*]: *exec message*

Configuration files

gridlabd.conf

- System copy is in **/usr/local/share/gridlabd** folder
- A local copy will override system copy
- Has system-wide settings
- Must be valid for all users

gridlabd-*user*.conf

- User-specific settings
- Based on USER or USERNAME environment variable
- Can be loaded **/usr/local/share/gridlabd** configuration file

Questions?

Getting Help

How to get help and report problems.

Help Resources

Runtime resources

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

Troubleshooting pages

- Documents warning and error messages

SF Wiki

- Latest documentation
- Searchable and you can make edits (registered users)

SF Forums

- Questions not covered by Wiki
- All users see answer

SF user emails

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

SF TRAC

- Used if none of the above addresses the problem
- Used to report "bugs" or requested upgrades

Command line: --help

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

Command-line options
--------------------
  --check|-c                              Performs module checks before starting simulation
  --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 and modules while simulation runs
  --quiet|-q                              Toggles suppression of all but error and fatal messages
  --verbose|-v                            Toggles output of verbose messages
  --warn|-w                               Toggles display of warning messages
  --workdir|-W                            Sets the working directory

Global and module control
-------------------------
  --define|-D <name>=[<module>:]<value>   Defines or sets a global (or module) variable
  --globals                               Displays a sorted list of all global variables
  --libinfo|-L <module>                   Displays information about a module

Information
-----------
  --copyright                             Displays copyright
  --license                               Displays the license agreement
  --version|-V                            Displays the version information
  --setup                                 Open simulation setup screen

...
```

Command line: --modhelp tape:player

```
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

```
host% gridlabd --modhelp powerflow:line
module powerflow {
    bool show_matrix_values;
    double primary_voltage_ratio;
    double nominal_frequency;
    bool require_voltage_control;
    double geographic_degree;
    complex fault_impedance;
    double warning_underfrequency;
    double warning_overfrequency;
    double warning_undervoltage;
    double warning_overvoltage;
    double warning_voltageangle;
    double maximum_voltage_error;
    enumeration {NR=2, GS=1, FBS=0} solver_method;
    bool line_capacitance;
    bool line_limits;
    char256 lu_solver;
    int64 NR_iteration_limit;
    int32 NR_superLU_procs;
    double default_maximum_voltage_error;
    double default_maximum_power_error;
    bool NR_admit_change;
    bool enable_subsecond_models; // Enable deltamode capabilities within the powerflow module
    bool all_powerflow_delta; // Forces all powerflow objects that are capable to participate in deltamode
    double deltamode_timestep[ns]; // Desired minimum timestep for deltamode-related simulations
    int64 deltamode_extra_function;
    double current_frequency[Hz]; // Current system-level frequency of the powerflow system
    bool master_frequency_update; // Tracking variable to see if an object has become the system frequency updater
    bool enable_frequency_dependence; // Flag to enable frequency-based variations in impedance values of lines and loads
    double default_resistance;
    bool enable_inrush; // Flag to enable in-rush calculations for lines and transformers in deltamode
    double low_voltage_impedance_level; // Lower limit of voltage (in per-unit) at which all load types are converted to impedance for 
in-rush calculations
    char1024 market_price_name;
}
...
```

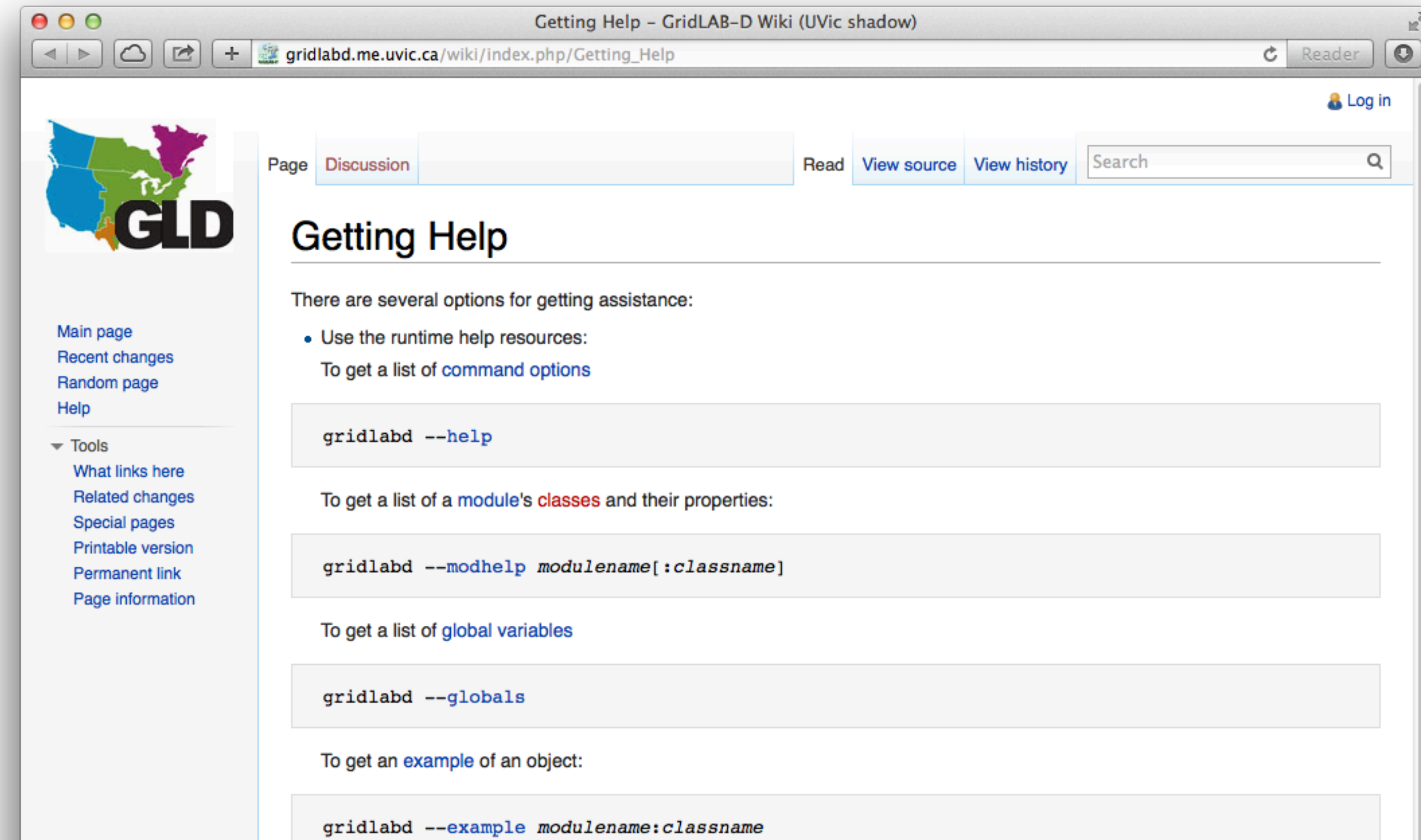
Command line: --modhelp

```
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)
          double power_out_real[W]; // power flow out (w.r.t to node), real
          complex power_losses[VA]; // power losses
          complex power_in_A[VA]; // power flow in (w.r.t from node), phase A
          complex power_in_B[VA]; // power flow in (w.r.t from node), phase B
          complex power_in_C[VA]; // power flow in (w.r.t from node), phase C
          complex power_out_A[VA]; // power flow out (w.r.t to node), phase A
          complex power_out_B[VA]; // power flow out (w.r.t to node), phase B
          complex power_out_C[VA]; // power flow out (w.r.t to node), phase C
          complex power_losses_A[VA]; // power losses, phase A
          complex power_losses_B[VA]; // power losses, phase B
          complex power_losses_C[VA]; // power losses, phase C
          complex current_out_A[A]; // current flow out of link (w.r.t. to node), phase A
          complex current_out_B[A]; // current flow out of link (w.r.t. to node), phase B
          complex current_out_C[A]; // current flow out of link (w.r.t. to node), phase C
          complex current_in_A[A]; // current flow to link (w.r.t from node), phase A
          complex current_in_B[A]; // current flow to link (w.r.t from node), phase B
          complex current_in_C[A]; // current flow to link (w.r.t from node), phase C
          complex fault_current_in_A[A]; // fault current flowing in, phase A
          complex fault_current_in_B[A]; // fault current flowing in, phase B
          complex fault_current_in_C[A]; // fault current flowing in, phase C
          complex fault_current_out_A[A]; // fault current flowing out, phase A
          complex fault_current_out_B[A]; // fault current flowing out, phase B
          complex fault_current_out_C[A]; // fault current flowing out, phase C
          set {CN=768, CR=512, CF=256, BN=48, BR=32, BF=16, AN=3, AR=2, AF=1, UNKNOWN=0} flow_direction; // flag used for describing direction of the flow of power
          double mean_repair_time[s]; // Time after a fault clears for the object to be back in service
          double continuous_rating[A]; // Continuous rating for this link object (set individual line segments
          double emergency_rating[A]; // Emergency rating for this link object (set individual line segments
          double inrush_convergence_value[V]; // Tolerance, as change in line voltage drop between iterations, for deltamode in-rush completion
     }

     function interupdate_pwr_object();
     function update_power_pwr_object();
     function check_limits_pwr_object();
     object configuration;
     double length[ft];
}
```

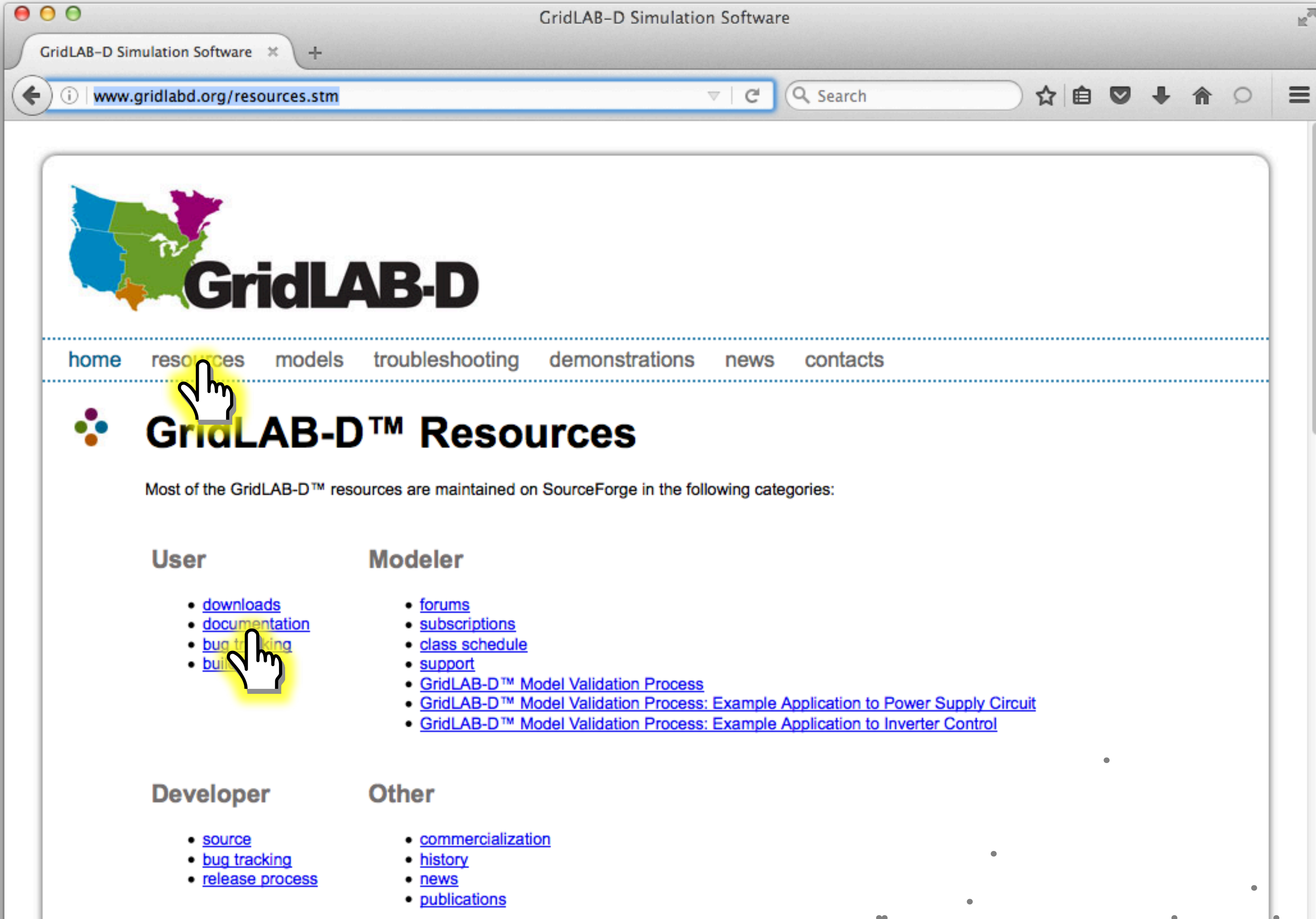
Command line: --info <topic>

Searches wiki pages for the topic

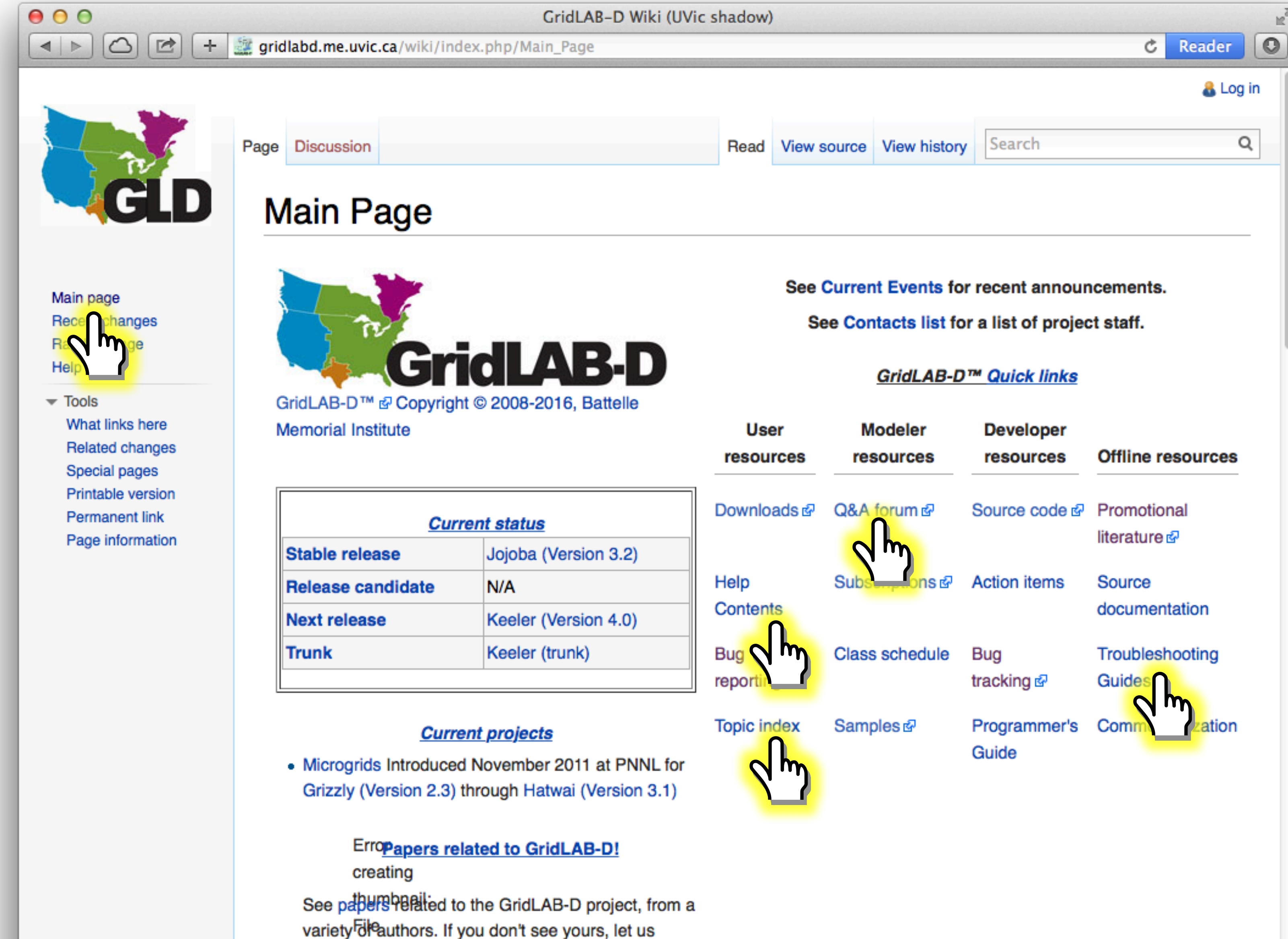
```
host% gridlabd --info "Getting help"
```



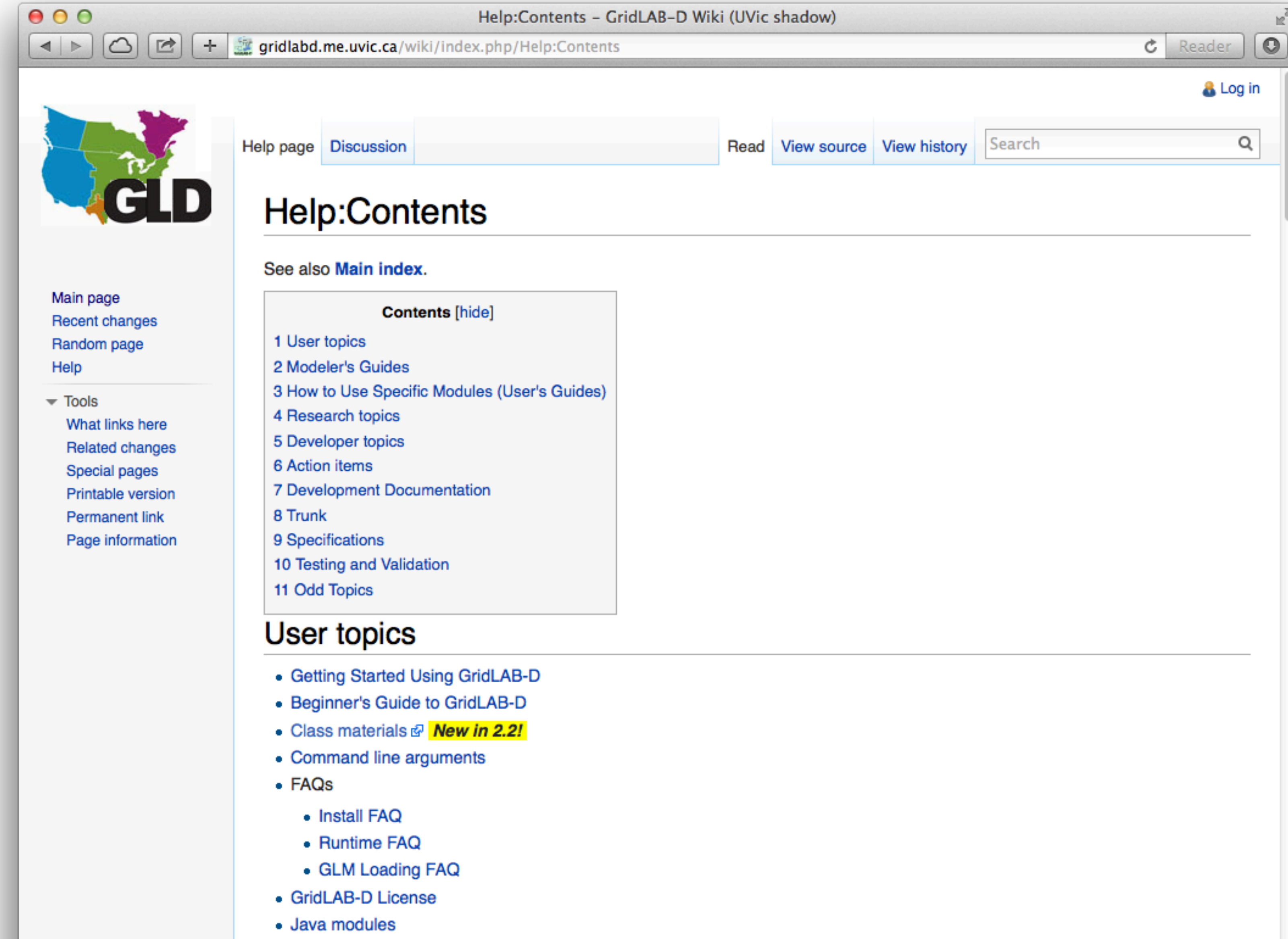
Online help resources



SourceForce MediaWiki

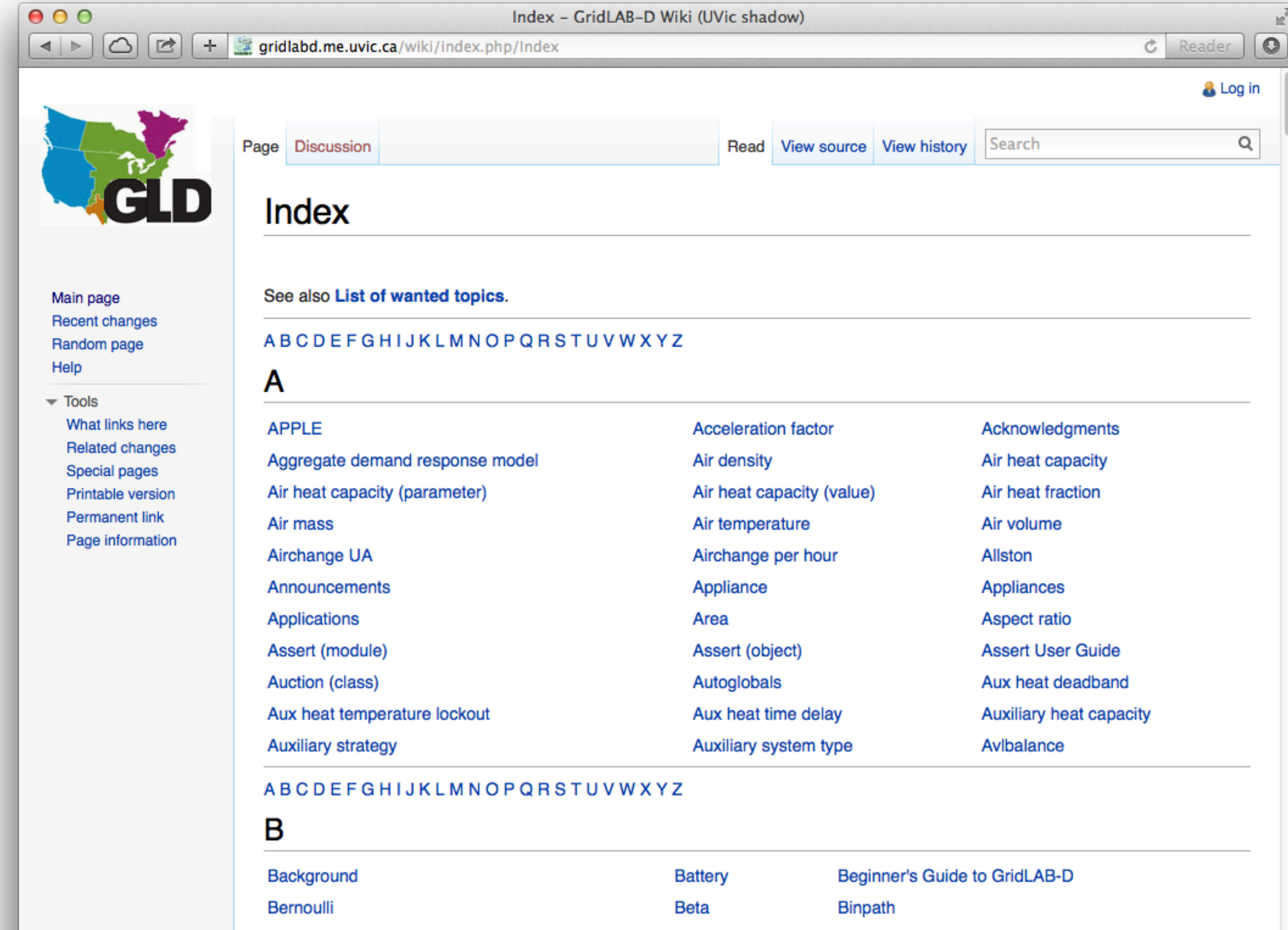


Help Guides

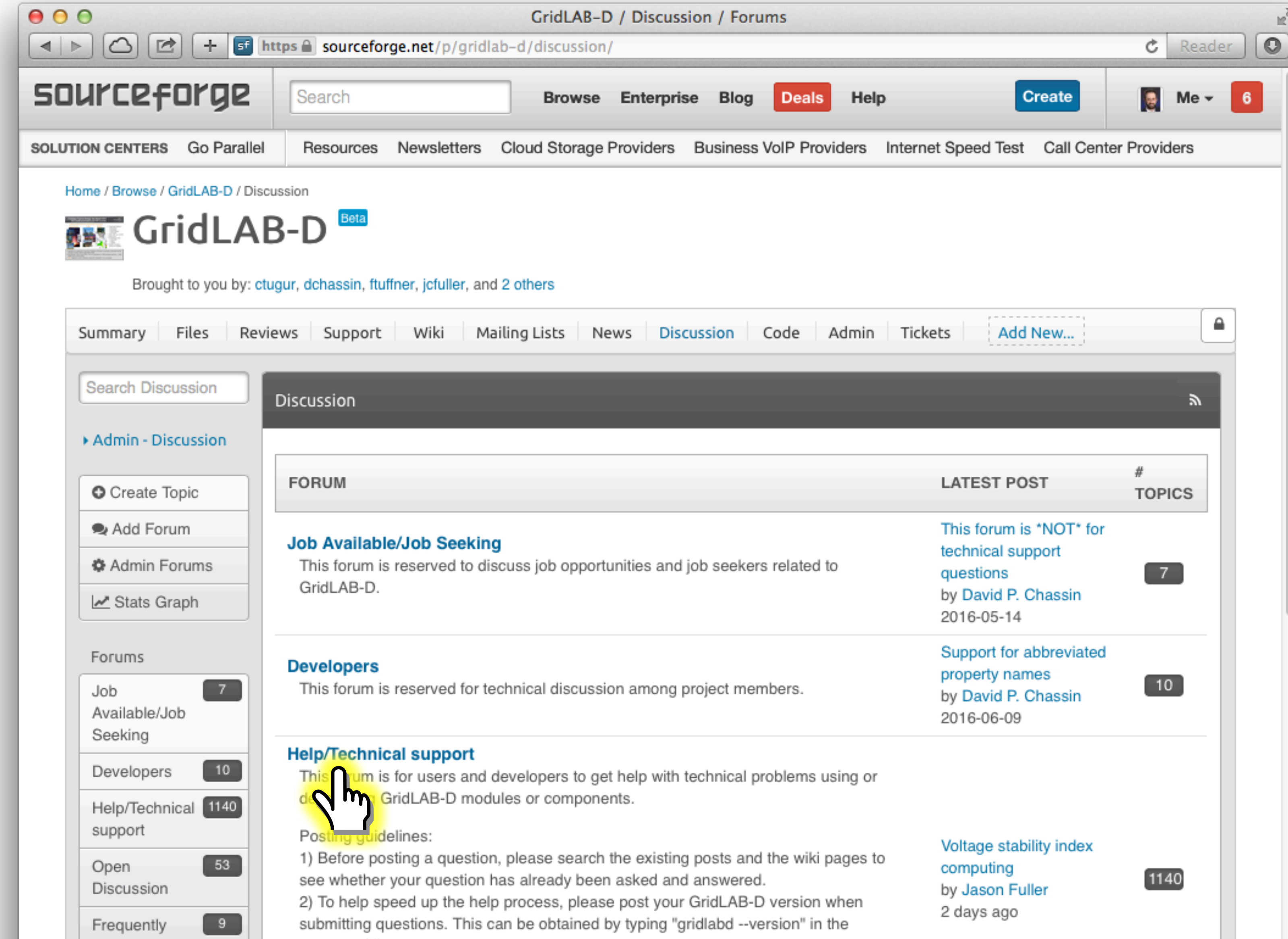


Help:Contents - GridLAB-D Wiki (UVic shadow)
gridlabd.me.uvic.ca/wiki/index.php/Help:Contents
Reader
Log in
GLD
Help page
Discussion
Read
View source
View history
Search
Help:Contents
See also Main index.
Contents [hide]
1 User topics
2 Modeler's Guides
3 How to Use Specific Modules (User's Guides)
4 Research topics
5 Developer topics
6 Action items
7 Development Documentation
8 Trunk
9 Specifications
10 Testing and Validation
11 Odd Topics
User topics
Getting Started Using GridLAB-D
Beginner's Guide to GridLAB-D
Class materials
New in 2.2!
Command line arguments
FAQs
Install FAQ
Runtime FAQ
GLM Loading FAQ
GridLAB-D License
Java modules
Main page
Recent changes
Random page
Help
Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

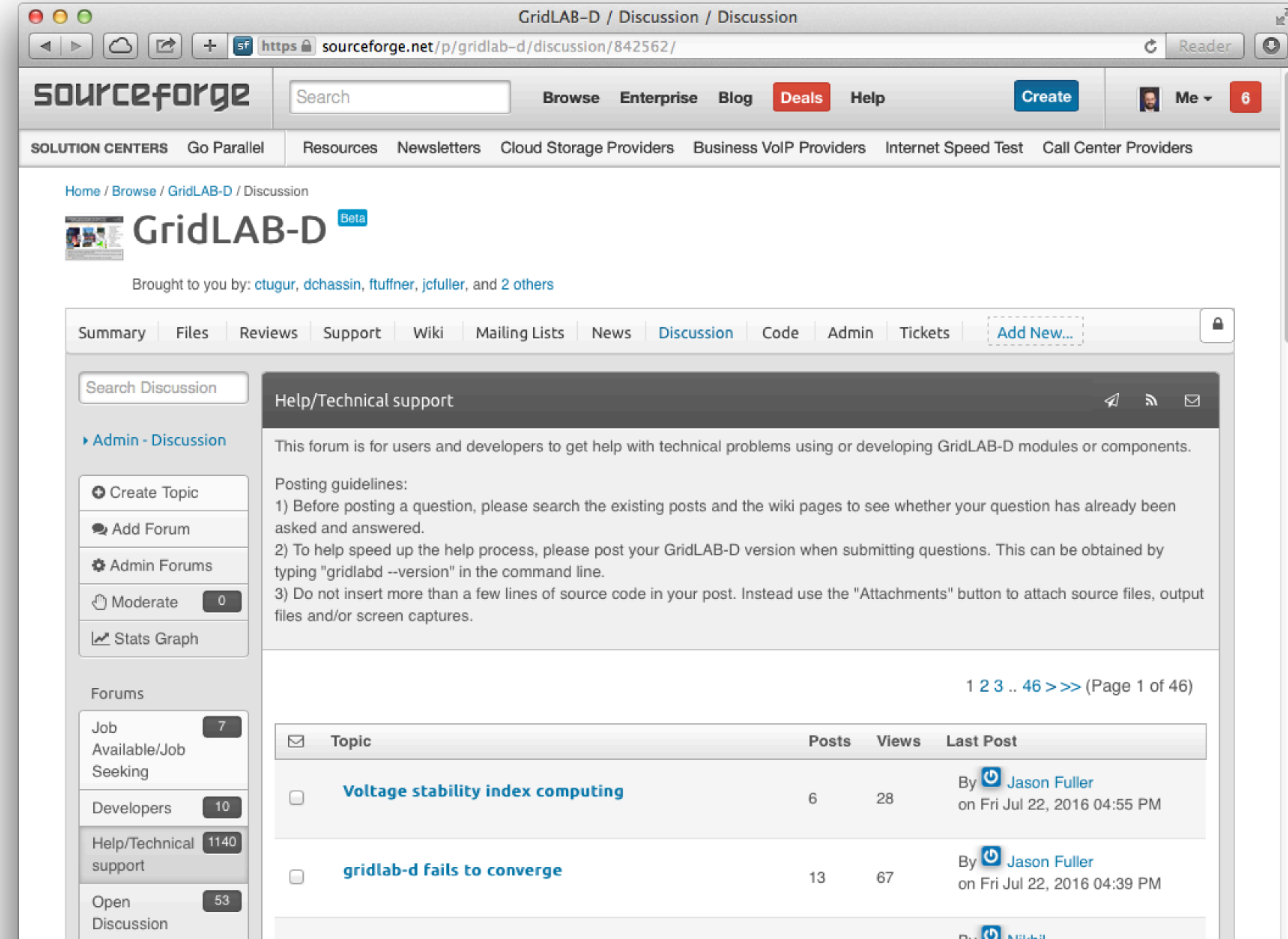
Topic Index


Index – GridLAB-D Wiki (UVic shadow)
gridlabd.me.uvic.ca/wiki/index.php/Index
Reader
Log in
GLD
Page
Discussion
Read
View source
View history
Search
Main page
Recent changes
Random page
Help
Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information
Index
See also List of wanted topics.
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A
APPLE
Acceleration factor
Acknowledgments
Aggregate demand response model
Air density
Air heat capacity
Air heat capacity (parameter)
Air heat capacity (value)
Air heat fraction
Air mass
Air temperature
Air volume
Airchange UA
Airchange per hour
Allston
Announcements
Appliance
Appliances
Applications
Area
Aspect ratio
Assert (module)
Assert (object)
Assert User Guide
Auction (class)
Autoglobals
Aux heat deadband
Aux heat temperature lockout
Aux heat time delay
Auxiliary heat capacity
Auxiliary strategy
Auxiliary system type
Avlbalance
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B
Background
Battery
Beginner's Guide to GridLAB-D
Bernoulli
Beta
Binpath

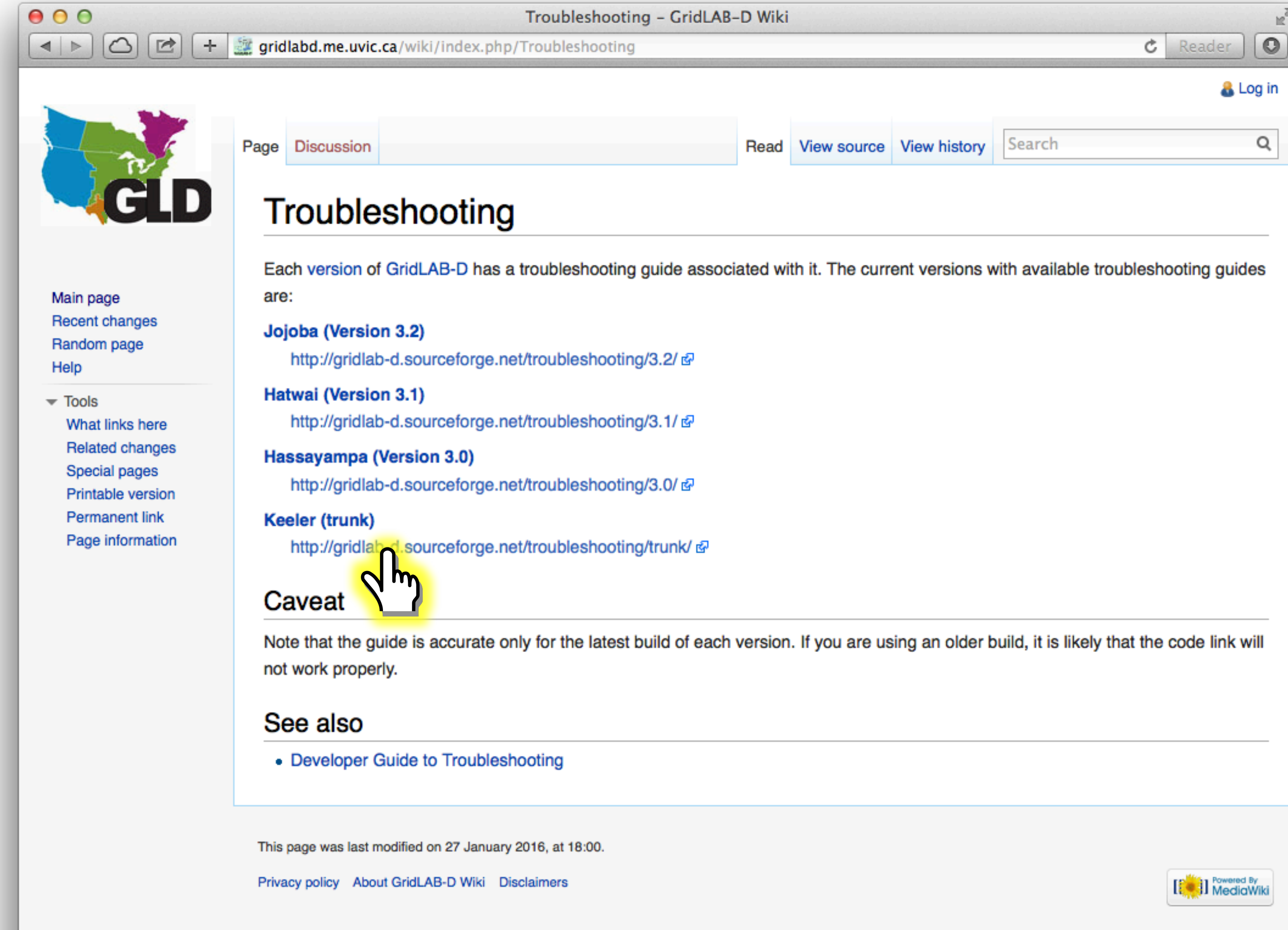
Forum


GridLAB-D / Discussion / Forums
https sourceforge.net/p/gridlab-d/discussion/
Reader
sourceforge
Search
Browse
Enterprise
Blog
Deals
Help
Create
Me
6
SOLUTION CENTERS
Go Parallel
Resources
Newsletters
Cloud Storage Providers
Business VoIP Providers
Internet Speed Test
Call Center Providers
Home / Browse / GridLAB-D / Discussion
GridLAB-D
Beta
Brought to you by: ctugur, dchassin, ftuffner, jcfuller, and 2 others
Summary
Files
Reviews
Support
Wiki
Mailing Lists
News
Discussion
Code
Admin
Tickets
Add New...
Search Discussion
Admin - Discussion
Create Topic
Add Forum
Admin Forums
Stats Graph
Forums
Job Available/Job Seeking 7
Developers 10
Help/Technical support 1140
Open Discussion 53
Frequently 9
Discussion
FORUM
LATEST POST
TOPICS
Job Available/Job Seeking
This forum is reserved to discuss job opportunities and job seekers related to GridLAB-D.
This forum is *NOT* for technical support questions
by David P. Chassin
2016-05-14
7
Developers
This forum is reserved for technical discussion among project members.
Support for abbreviated property names
by David P. Chassin
2016-06-09
10
Help/Technical support
GridLAB-D modules or components.
1) Before posting a question, please search the existing posts and the wiki pages to see whether your question has already been asked and answered.
2) To help speed up the help process, please post your GridLAB-D version when submitting questions. This can be obtained by typing "gridlabd --version" in the
Voltage stability index computing
by Jason Fuller
2 days ago
1140

Posting to the forum



Troubleshooting



Troubleshooting

gridlab-d.sourceforge.net/troubleshooting/3.2/

This troubleshooting guide lists all the errors and warning messages from GridLAB-D. Simply search for your message and follow the recommendations given. *Last updated Fri Jun 26 02:30:48 PDT 2015*.

Warnings | Errors | Fatal errors | Exceptions

Warnings

!	\	A	B	C	D	E	F	G	I	L	M	N	O	P	R	S	T	U	V	W	Z

'*string*' ('*string*':'*integer*') is triplex but doesn't have phases S set

- A triplex line has been used, but this triplex line does not have the phase S designated to indicate it contains single-phase components. Without this specified, you may get invalid results. *See powerflow/triplex_line.cpp(76)*.

'*string*' ('*string*':'*integer*') parent '*string*' ('*string*':'*integer*') does not export attach_enduse function so voltage response cannot be modeled

- Enduses must have a voltage source from a parent object that exports an attach_enduse function. The residential_enduse object references a parent object that does not conform with this requirement. Fix the parent reference and try again. *See residential/residential_enduse.cpp(85)*.

'*string*' (auction:'*integer*') capacity_reference_property '*string*' is not a double type property

- capacity_reference_property must specify a double property to work properly, and may behave unpredictably should it be pointed at non-double properties. *See market/auction.cpp(345)*.

'*string*' (id:'*integer*') already has a link parented - this could give invalid answers in FBS!

- While connecting the different objects, a link's 'to' end already has another link as a parent. This is indicative of a loop or other 'non-radial' condition on the system. The exception to this is if an open switch is connected at that node. If it is not an open switch, Forward Backward-Sweep (FBS) will not give a proper solution in this condition. Even if it is an open switch, be aware that FBS could still give an invalid answer, depending on topological conditions. If it is an open switch, consider removing the open switch from your GLM to ensure proper answers. If it is another type of link or you do not wish to remove the open switch, consider using the Newton-Raphson (NR) solver instead. *See powerflow/link.cpp(305)*.

Questions?

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Modeling language basics

Detailed discussion of GridLAB-D modeling language (GLM).

Top-level Directives

Instructs loader on the content of the next GLM block

- **clock** – controls time
- **module** – loads groups of classes and functions
- **class** – defines a class
- **object** – defines an object
- **import** – imports a model
- **export** – exports a model
- **link** – links a model to an external tool
- **filter** – defines a filter to connect object properties
- **extern** – defines a function to connect object properties

Clock directive

```
clock {
      timezone PST+8PDT;
      starttime '2001-01-23 01:23:45 PST';
      stoptime '2001-06-12 01:23:45 PDT';
}
```

Specifies the timezone (must come first)

Sets the start time

Sets the end time (if steady state not reached first)

Module directive

```
module powerflow {
      solver_method NR;
      NR_iteration_limit 50;
}
```

Loads the named module

Sets the module's variables (overrides default)

Modules define one or more classes

Modules often come with other functions

Class directive

```
class house {
      double floor_area[sf];
}
```

Asserts structure of an existing class

- Load fails if actual structure does not match

Add new variables to an existing class

- Type, name and units are defined.

Can be used to create a new class

- Must include **init** and **sync** (more on this later)
- (Windows users must installed **mingw**)

Property types

bool

char8, char32, char256, char1024

int8, int16, int32, int64, timestamp,

enumeration, set

double, complex, float, real, randomvar, double_array,

loadshape, enduse, object

Example property values

```
is_enabled TRUE; // boolean
motor_state STALLED; // set/enumeration
city "Richland"; //char*
start_at '2008-04-01 12:00:00 PDT'; // timestamp
floor_area 2000; // double
floor_area 250 m^2; // double with units
floor_area random.triangle(1500,2500); // functional
wall_area (sqrt($var)*8); // calculated
name `{class}:{id}`; // expansion
name ${var++}; // expression
test_result (test_value) ? "<=0" : ">0"; // trinary
```

Property units

Defines how values are stored in memory

- Double and complex converted automatically

General unit conversion system used

- Standard units (m, W, V, A, h, s)
- Conventional units (kW, kVA, Btu)
- Derived units (e.g., MW/h)—beware of quirks
- Complete list of units in **unitfile.txt**
- If unit is not define the value is dimensionless
- If unit is not specified the property's unit is assumed

Reserved property names

Used by object headers

Cannot be used in class definitions

```
class invalid {
        object parent;
}
```

Can be set in object definitions

```
object dryer {
        parent my_house;
}
```

- ✓parent
- ✓rank
- ✓clock
- ✓valid_to
- ✓latitude
- ✓longitude
- ✓in_svc
- ✓out_svc
- ✓flags

Object directive

Basic syntax

```
object classname {
  property_name value;
  ...
}
```

Defining multiple objects at once

```
object class:..count ...
```

Numerical identifiers

```
object class:num ...
object class:first..last ...
```

Rules for values

Double quotes required if string has spaces

- Non-string values are converted automatically
- Quotes suppress all other rules (math, expansions, etc.)

Times and other special types

- Requires single quotes when spaces are included

Functional values allowed outside parentheses

- `random.dist(args...)`

Some properties use extended syntax, e.g.,

- `my_randomvar {type:normal(0,1); min:0; refresh:30s; };`

Expression syntax

Standard math operations used

```
((12+8)/9)
```

Usual math functions supported

```
(sin((12+8)/2*Pi))
```

Can refer to other properties of object, e.g.,

```
(… $property_name …)
(… this.property_name …)
(… parent.property_name …)
(… object.property_name …) ← caveat!
```

Functionals

Only for random numbers

Distributions require arguments

Discrete distributions

- `degenerate`(x)
- `bernoulli`(p)
- `sampled`$(n, x_1, x_2, \ldots, x_n)$

Others likely coming

Continuous distributions

- `uniform`(a, b)
- `normal`(m, s)
- `lognormal`(m, s)
- `exponential`(l)
- `pareto`(m, g)
- `beta`(a, b)
- `gamma`(a, b)
- `weibull`(l, k)
- `rayleigh`(s)
- `triangle`(a, b)

Nested objects

Implicit parent-child relationship

Allows generation of population

Avoid having to provide names

Equivalent:

```
object house {
  object dryer {
    ...
  };
  ...
}
```

=

```
object house {
name my_house;
...
    }
    object dryer {
parent my_house;
...
}
```

No equivalent: `object house:..10`

Macro processing

Hash (#) in first non-white of a line specifies macros

Macro processing is done first

New values stored as global variables

```
#define MYDEF=text
```

Inline substitutions of macros syntax

```
${MYDEF}
```

Environment variables also substituted

```
${USER}
```

Standard macros

Standard expressions

- `#include` *filename*
- `#print` *expression*
- `#warning` *message*
- `#error` *message*
- `#define` *name*=*value*
- `#set` *name*=*value*
- `#setenv` *name*=*value*

Conditionals

- `#if` *expression*
- `#else`
- `#endif`
- `#ifdef` *name*

  …
- `#ifndef` *name*

  …
- `#ifexist` *file*

  …

Schedules, Loadshapes, and Enduses

***schedule* directive is available**

- Provides efficient built-in support for periodic values

Loadshape property

- Reads a schedule to update load
- Implements various typical load behaviors

Enduse property

- Reads loadshape
- Synthesize power demand

Questions?

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Data Input and Output

Introduce objects that control flow of data in and out of GridLAB-D.

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Extracting final results: XML dump

XML files describe models

- Classes, objects, players, recorders and clocks

Strengths

- Highly compatible with other software
- Faithful representation of a single model instance

Weaknesses

- Viewing and editing only with third-party tools
- Rigid (non-parametric) syntax
- Not good for creating large complex models
- No populations or distribution allowed
- Not easily used for extraction of time series data

Extracting intermediate results

Recorders

- Samples a single object's properties at designated times
- Handles unit conversion if unit is specified
- Samples stored in file or database (depends on module)
- Triggers can start recording conditionally
- Limits can end recording conditionally

Multi-recorder

- Multi-object variant records several objects

Collector

- Aggregates samples (e.g., mean, stdev, min, max, etc.)

Histogram

- Counts samples in bins, saves and reset at intervals

Introducing data

Players

- Alter object properties at designated times
- Used to alter boundary conditions
- Can use relative time and looped

Shapers

- Schedules (using POSIX *cron* standard)
- Amplitude: copies value directly
- Pulse-width: on/off with a probability
- Queues: accrues to threshold before “on”

Tape Module

Changing properties

- Player
- Shaper

Sampling properties

- Recorder
- Multi_recorder
- Collector
- Histogram
- Group_recorder
- Violation_recorder

MySQL Module

- Database

Mimics tape module

- Player
- Recorder
- Collector

Internal data sources and links

Schedules

- Represent periodic data (minute, hour, day, month, weekday)
- More efficient than player or shaper objects
- Preloaded into core memory on initialization
- Often used in conjunction with **loadshape** and **enduse**

Transforms

- Defines reusable functions to transform a source property and apply it to a destination property

Filter

- Defines a reusable discrete-time digital filter to transform a source property and apply the result to a destination property

Summary

Player and Shaper

- Used to update information at a specific time from a file

Recorder and Collector

- Used to collect information from the model
- Can also plot to screen or file

Loadshapes and Schedules

- NOT part of the tape module, but similar functionality
- Updates recurring information at a specific time directly from the core (not from a file)

Transforms and Filters

- Uses data from a source variable to modify other properties

Players

Objects that set and control the boundary conditions.

Input object: player

Implemented in tape and mysql modules

- One of three primary ways of inputting data
- Tapes apply a time-series to a single property of the parent object

Several possible sources of tape data

- File : source is a CSV files
- ODBC : source is a database
- Memory : source is a global variable
- MySQL : source is a table

Player timestamps

Timestamp is the beginning of interval

- Absolute time uses ISO time format

  YYYY-MM-DD hh:mm:ss [ZZZ]
- Missing timezone implies UTC (not local timezone!)
- Relative times use "+" prefix, e.g., "+600"
- Relative times can have units (default is seconds)

  "s", "m", "h", "d", "w",

  Examples: +12h

<u>Warning</u>: "m" is not the standard unit for minutes. The tape modules does not use GridLAB-D's unit conversion system, in which the standard unit for a minute is "min".

Player "property"

Name of parent's property that is updated

Data conversion is automatic

- Conversion is from text input
- Boolean uses TRUE or FALSE and/or 0 or 1
- Integer uses atoi()
- Enumeration/set uses keywords (specific to property)
- Double uses **double_format** (a global variable)
- Complex uses **complex_format** (a global variable)
- Object references use **object_scan** format (global)

<u>Warning</u>: Tape player does not support units but mysql player does.

Player “file”

Determines the source CSV file of the data

- Require that **file** property be the filename
- Format is CSV, but others may be coming

Directory delimiter can be system specific

- “/” are normally used; “\” allowed in Windows

Meaning differs based on filetype property

- Default is “txt”
- “odbc” is for databases: **file** is connection string
- “memory” is for Matlab data: **file** is global variable name
- Not supported in mysql player

Player “loop”

Indicates how many times the data source is to be read

- Default (0) is to not loop

Absolute timestamps are read only on first pass

Relative time offsets are used on all passes

Example player object

```
object house {
        floor_area random.normal(1500,300);
        object player {
                property air_temperature;
                file "tair.csv"
                loop 10;
        };
}
```

Example player file

```
# comment lines start with hash sign
# this example will produce a square wave timeseries
# alternating every hour between 72 and 73

# this is an absolute timestamp
# it is read only on the first pass
2007-01-01 00:00:00, 72.0

# these are relative timestamps
# they are read on every pass
+1h, 73.0
+1h, 72.0
```

Another Example

Another way to code

```
object house:1 {
        floor_area random.normal(1500,300);
}

object player {
        parent house:1
        property air_temperature
        file "tair.csv";
        loop 21;
}
```

Yet another Example

Another way to code

```
object house {
        name MyHouse1;
        floor_area random.normal(1500,300);
}

object player {
        parent MyHouse1;
        property air_temperature
        file "tair.csv";
        loop 21;
}
```

Another example player file

```
# comment lines start with hash sign
# this example will produce a time series of
# temperatures for midnight-3 am, then stay at
# 68.0 degrees indefinitely
2007-01-01 00:00:00, 72.0
2007-01-01 01:00:00, 73.0
2007-01-01 02:00:00, 72.0
2007-01-01 03:00:00, 68.0
```

Demo

Manipulate thermostatic controls using player: *player.glm*

```
clock {
      timezone PST+8PDT;
      starttime '2001-01-01 00:00:00 PST';
}

module residential;
module tape;

object house {
       heating_setpoint 40 degF;
       cooling_setpoint 90 degF;
       object player {
              property cooling_setpoint;
              file theat.csv;
              loop 28;
       };
}
```

Demo

Another way to manipulate values using player: player_2.glm

```
clock {
      timezone PST+8PDT;
      starttime '2001-01-01 00:00:00 PST';
}

module residential;
module tape;

object house {
       name MyHouse1;
       heating_setpoint 40 degF;
       cooling_setpoint 90 degF;
}
object player {
       parent MyHouse1;
       property cooling_setpoint;
       file theat.csv;
       loop 28;
}
```

Example player file: theat.csv

```
2001-01-02 06:00:00,75
+15h,60
+9h,75
```

Note: Excel does not handle ISO date formats, timezones, or daylight savings time rules correctly. To avoid issues, use UTC time for all timestamps as much as possible.

Tip: In Excel use custom cell formats for timestamps:

```
yyyy-mm-dd HH:MM:SS
yyyy-mm-dd HH:MM:SS "PST"
```

Player transforms

```
object class {
        property source*scalar+offset;
}
```

Used to create a linear function of value

- y = m*x + b
- A single common input used by multiple objects

Easy offsets and scaling of shapes from a single source

Player transforms

SLAC

Created in multiple steps:

```
// trick to publish the input value when no parent is set
class player { // this only needs to be done once
  double value; // at the beginning of the glm file
}
object player { // this is only done once for each input
  name THeat;
  file theat.csv;
  loop 5;
}
object house:1 {
  heating_setpoint THeat.value*1+2;  // syntax and order important
}
object house:2 {
  heating_setpoint THeat.value*1.1+4;  // using different offset/scale
}
```

This input can now be used by many objects

ODBC Tapes

ODBC data for players and recorders only

```
file odbc:[DSN][:uname:pwd]:[objName];
file odbc:MyDataSource:user:secret:MyHouse1;
```

- Keys entries by tape name and object name
- Expects fixed table names, replaces same-key objects each run

<u>Note</u>: multiple properties are processed together as a single "column"

ODBC Tape Tables

EVENT_TABLE and OBJECT_TABLE	
EVENT_TIME	TIMESTAMP
EVENT_VAL	String
EVENT_OBJECT_NAME	Char32
EVENT_LINE	Integer

- OBJECT_TABLE writes recorder output.
- EVENT_TABLE reads player input.
- EVENT_LINE records the order in which lines are to be read.
- HEADER_TABLE is only written to. Only HEADER_OBJECT_NAME is referenced by the system, and writes the recorder's name.

HEADER_TABLE	
HEADER_OBJECT_NAME	String
HEADER_TIME	String
HEADER_USERNAME	String
HEADER_HOSTNAME	String
HEADER_TARGET	String
HEADER_PROPERTY	String
HEADER_INTERVAL	Long
HEADER_LIMIT	Long

Time, username, hostname, target, property, interval, and limit all correspond with the CSV file header output.

MySQL player

Very similar to tape player objects

Requires database object to operate

- If none use, the default database is used
- Default database is specified by module declarations

```
module mysql {
        hostname "localhost";
        username "gridlabd";
        password "";
        schema "gridlabd";
        port 3306;
        socketname "/tmp/mysql.sock";
        clientflags
            COMPRESS|FOUND_ROWS|IGNORE_SIGPIPE|INTERAC
            TIVE|LOCAL_FILES|MULTI_RESULTS|MULTI_STATE
            MENTS|NO_SCHEMA|ODBC|SSL|REMEMBER_OPTIONS;
}
```

MySQL database

```
object database {
        hostname "localhost";
        username "gridlabd";
        password "";
        schema "gridlabd";
        port 3306;
        socketname "/tmp/mysql.sock";
        clientflags
            COMPRESS|FOUND_ROWS|IGNORE_SIGPIPE|INTERACTIVE|LOC
            AL_FILES|MULTI_RESULTS|MULTI_STATEMENTS|NO_SCHEMA|
            ODBC|SSL|REMEMBER_OPTIONS;
        options SHOWQUERY|NOCREATE|NEWDB|OVERWRITE;
        on_init sql-script-name;
        on_sync sql-script-name;
        on_term sql-script-name;
        sync_interval seconds;
}
```

MySQL player

```
object player {
        property property-name;
        table|file source-table;
        mode {"r","r+"};
        filetype {"CSV"};
        connection database-object-name;
        options 0;
        loop number-of-loops;
}
```

Questions?

Recorders

This section will introduce you to objects that record data in other objects.

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Output object

Implemented in tape and mysql module

- One of two primary ways of collecting data
- One of few objects which “drive” the simulation
- Recorders observe one or more properties of a single object

Several possible destinations for data (tape)

- File : source is a specially formatted files
- ODBC : source is a database
- Memory : source is a global variable
- Plot : gnuplot output

Property

Name of parent's property that is recorded

Internal data conversion is performed

- Conversion is to text output
- Booleans uses TRUE or FALSE and/or 0 or 1
- Integers use decimal integer
- Enumerations and set use keywords
- Doubles use **double_format**
  - `#set double_format=%.12lg` *allows user to modify format*
- Complex uses **complex_format**
- Object references use **object_format**

Property (cont.)

Unit conversion only performed if unit is specified

```
property name[unit];
```

- Example:

```
object house {
        floor_area random.triable(1000,2000) sf;
        object recorder {
                property floor_area[m^2];
                // ...
        };
}
```

Additional extensions used for complex numbers

- real, imag, mag, ang, arg

```
property voltage_A.real;
```

File

Meaning differs based on filetype property

- Format can be system specific

  "/" are normally used; "\" allowed in Windows
- File must be writeable
- Path to file is not automatically created
- Existing files are overwritten

<u>Note</u>: Write failure is not an error

(simulation continues with a warning)

Interval

Determines the sampling interval for data

- How often should I sample?

Units are seconds

- -1 means sample transients (on change)
- 0 means sample each iteration

Interval “drives” simulation

- Affects numerical results if models don’t handle transients well

Note: recording samples are lagging
(unlike players, which are leading)

Limit

The maximum number of samples

- How many samples should I record?

Limits the size of the output file

- 0 is default
- 0 means no limit

<u>Note</u>: If a “stoptime” is not specified, simulation runs until all recorders reach their limits.

Trigger

Specifies condition to start recording

- Works only for the target property
- Usual compare operations apply
- Once triggered, recording continues to limit

Example

```
trigger "< 0"; // start recording when target is negative
```

Multi-Recorder

Same basic syntax as a recorder

- Allows user to record from multiple objects into a single file

Example:

```
object meter {
        name meter1;
        phases ABCN;
        nominal_voltage 7200;
        object multi_recorder {
                property "measured_real_power,meter2:constant_power_A_real";
                limit 8808;
                interval 1800;
                file record_power.csv;
        };
}
```

non-parent object

Demo

```
// player_recorder.glm
module residential;
module tape;
clock {
      timezone PST+8PDT;
      starttime '2001-01-01 00:00:00 PST';
}
object house:1 {
       heating_setpoint 40degF;
       cooling_setpoint 90degF;
}
object player {
       parent house:1;
       property cooling_setpoint;
       file theat.csv;
       loop 100; // does not drive simulation
}
object recorder{
       parent house:1;
       property air_temperature,cooling_setpoint;
       file theat_record.csv;
       interval 7200; // 2 hours
       limit 48; // records 48 samples -> drives simulation for 96 hours
}
```

Demo

```
// player_recorder_interval.glm
module residential;
module tape;
clock {
      timezone PST+8PDT;
      starttime '2001-01-01 0:00:00 PST';
}
object house:1 {
         heating_setpoint 40degF;
         cooling_setpoint 90degF;
}
object player {
         parent house:1;
         property cooling_setpoint;
         file theat.csv;
         loop 100;
}
object recorder{
         parent house:1;
         property air_temperature,cooling_setpoint;
         file theat_record_0.csv;
         interval 0; // sample each iteration (9pm and 6am from player)
         limit 48;
}
```

Example: intervals

```
// Interval -1, 0, 7200: player_recorder_trio.glm
object player {
        parent house:1;
        property cooling_setpoint;
        file theat.csv;
        loop 100;
}
object recorder{
        parent house:1;
        property air_temperature,cooling_setpoint;
        file theat_record_7200.csv;
        interval 7200; // records every 2 hours for 48 x 2 hours
        limit 48;
}
object recorder{
        parent house:1;
        property air_temperature,cooling_setpoint;
        file theat_record_0.csv;
        interval 0; // records every iteration (9pm and 6am from player)
        limit 48;
}
object recorder{
        parent house:1;
        property air_temperature,cooling_setpoint;
        file theat_record_1.csv;
        interval -1; // record every change in value when clock advances
        limit 48;
}
```

Questions?

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Collectors

Objects that record aggregate data from groups of objects.

Output object

Implemented in tape and mysql module

- One of two primary ways of recording data
- Same basic parameters as recorders

Collectors take aggregates of properties

- Observations collected from a group of objects
- Aggregations include statistics and min/max
- Aggregations are specified as part of property

Group

Defines the subset of objects to observe

- Object properties used to define group
- Header properties allowed:

```
class, parent, rank, in_svc, out_svc, groupid
```

Search is performed only on first observation

- Search result is reused thereafter
- => Groups must be constant over time
- => Only time-invariant properties may be used in group criteria

Example: `group "class=house";`

- Can include multiple groupings

```
group "class=house AND groupid=feeder1"
```

Property aggregators

Similar to recorders, but with aggregators

- count, min, max, avg, std, sum, prod
- mean, var, kur
- gamma

Parts used for complex values

- real, imag
- mag, ang, arg

Example: `property "sum(power.mag)"`

Example

```
object collector {
        group “class=house AND groupid=feeder1”;
        property “sum(power.mag),avg(hvac_load.real)”;
        interval 3600;
        limit 24;
}
```

Demo

```
// demo/collector.glm sum and average of the real power of light objects in houses
module residential{
       implicit_enduses NONE;
}
module tape;
clock {
      timezone PST+8PDT;
      starttime '2001-01-01 0:00:00 PST';
      stoptime '2001-07-01 00:00:00 PST';
}
schedule light_demand {
         * 1-3 * * * 0;
         * 4-6 * * * 0.15;
         * 7-19 * * * 0;
         * 20-0 * * * .85;
}
object house:..10 {
       cooling_setpoint 90 degF;
       object lights {
              shape "type: analog; schedule: light_demand; power: 1.1 kW";
       };
}
object collector {
       file theat_collector.csv;
       group "class=lights"; // sample only lights
       property sum(energy.real),avg(energy.real);  // use real part--aggregators need doubles
       interval 3600;
       limit 744;
}
```

Questions?

Exercises (players)

1) Create a player file that can be used to inject a daily heating setpoint. Nightime (72F) is 9PM to 6AM and daytime (75F) is 6AM to 9PM. Hint: use relative times.
2) Attach the player to a house and run the simulation for 4 weeks. Hint: you will need to loop the player file.
3) Modify Exercise 2 to create an additional home that uses a player transform to inject a daily heating setpoint. Nighttime (74F) is 9PM to 6AM and daytime (77F) is 6AM to 9PM.

Exercises (recorders)

4) Create a recorder to collect hourly indoor air temperature of the default house for the year 2001.
5) Create a recorder to collect 100 values of the default waterheater power transitions (actual_load) in the default house.
6) Create a multi-recorder to record the waterheater power and the house HVAC power (hvac_load) at five minute intervals for one week.
7) Create two recorders to collect voltages in the IEEE 37 node test model (use load701 & node775; voltage_A, voltage_B, voltage_C) to observe the voltages as they converge. Hint: when using Excel, *.real and *.imag are very helpful.

Exercises (collectors)

8) Collect total power consumption for a population of 100 water heaters for one month. Hint: use “actual_load”.
9) Collect the hourly mean, standard deviation, minimum and maximum indoor air temperature a population of 100 default houses for a week. Hint: use “air_temperature”.
10) Collect the minimum and maximum voltages of all nodes in the IEEE 37-bus test model as it converges. Hint: use voltage_A.mag, voltage_B.mag, & voltage_C.mag