Users Guide for the **StaticCapPlan** and **UnitCommit** Models

With brief notes about the stochastic versions: StocCapPlan and StocUC

Advanced-Power Model Family
Release 3 (SVN ver526)

by
Bryan Palmintier
MIT
December 2012

1 Overview

`StaticCapPlan` and `UnitCommit` are modular, highly configurable GAMS based electric power system models. They form the core of the "AdvancedPower" family of models. `UnitCommit` is a unit commitment model that is capable of both traditional individual unit binary commitment and the more efficient clustered integer unit commitment formulation developed in my dissertation. `StaticCapPlan` is a single period generation expansion planning model that uses `UnitCommit` to compute operations costs.

The high degree of configurability allows both models to be used for everything from very simple merit order analysis up to highly constrained unit commitment simulation complete with minimum up and down time, multiple reserve classes, multi-segment piecewise linear fuel use curves (aka heat rate curves or cost curves), etc. The primary limitation in the current version of the models is that they ignore the transmission system and assume all generators and all loads are connected at a single location.

The configurability comes from a combination of a rich set of command-line options, the ability to define partially to fully customized data and model include files, support for "update" files that automatically override a subset of the data files, and for those familiar with GAMS the ability to add additional model equations. The models consist of the core model files, helper utilities, and a collection of "shared" (between the models and by other similar models) model pieces. In addition to the core model and associated model pieces, both models also use a common set of data files for system parameters, demand time series, generator data, fuel parameters, and generator availability (including wind output). Note that since these separate model pieces are organized across multiple files and directories, copying and using the model requires copying the entire directory structure, rather than a single GAMS model file.

2 Table of Contents

3 Basic Use: StaticCapPlan

The simplest capacity planning call is:

```
gams StaticCapPlan
```

This is normally run from the …/models/capplan directory[1]. It should only take a few seconds (or less) and conducts a simple capacity plan using the test*.inc set of data. These and other data files are found in the …/models/data directory. The results will be stored as tables in a set of comma separated text files (*.csv) as described below.

3.1 Basic StaticCapPlan Results

These show up in …/models/capplan/out/ after a successful model run
[Use --out_dir=DIR to change the path relative to the model file. Default is out/]

- **SCP_summary.csv:** Contains a detailed summary of the results (e.g. cost breakdown, emissions by type, etc.), configuration (e.g. input data files, option flag settings, etc.), and run information (e.g. solver time, number of equations, etc.) In many cases this will contain all of the output data you need. It is a strictly two column format with separate field name and values. The fieldnames include units and should be self describing. Only alphanumeric symbols plus the underscore ("_") are used for fieldnames so they can readily be used as variable names, structure field ids, or dictionary/hash keys for an automated parsing programs, yet the names are also descriptive enough to be human readible.
- **Other outputs:** Complete build summary, capacity summary, dispatch time series, and other files are also created in the same directory. [You can suppress printing these using the --summary_only=1 option] All of the output files share the same prefix. [Defaults to "SCP_" Can be changed using --out_prefix=NEW_PREFIX_].

3.2 Basic Features (StaticCapPlan)

The basic, no frills plan uses

- *Integer build decisions:* That is investments must be made in multiples of the notional gen_size parameter [specified in either the test_gen.inc input file]. This makes for a simple Mixed-Integer problem (MIP)[for continuous build decisions use the --ignore_integer=1 option];
- *Allowed non-served energy:* When operational costs are too high, or constraints are too strict, load is shed for a price, specified by pPriceNonServed in the system definition file. [Use --no_nse=1 to disable non-served energy and force supply to equal demand, The cost of non-served energy is specified in the test_sys.inc file.];
- *Allowed wind/renewable shedding:* If/when cost effective, excess wind is shed (not used). This setting typically has no effect unless reserves, unit_min, or unit commitment constraints are used. [Use --force_renewables=1 to disable shedding. Disabling shedding only is possible for lower quantities of renewables]; and

[1] These directions assume running from the command line. When running using the Windoze IDE, setup a new project in this same directory, open StaticCapPlan.gms and run as usual. In more complicated examples, the command line options can be entered into the command line option box near the top right of the IDE, just below the menu bar.

- *Automatic anualization of capital costs:* The capital cost, c_cap, is specified in actual total capital cost and automatically annualized by the model using the capital recovery factor based on the system weighted average cost of capital (WACC) [specified in `test_sys.inc`] and the plant's life [specified in `test_gen.inc`].

These basic features represent only a smalls subset of StaticCapPlan's features. Section 5 describes a more complex example and section 11 describes the complete set of configuration options.

4 Basic Use: UnitCommit

The simplest unit commitment call is:

```
gams UnitCommit
```

This is normally run from the …/models/ops directory[2]. It should only take a few seconds (or less) and conducts a simple unit commitment plan using the test*.inc set of data[3]. Similar to the capacity planning model description above, UnitCommit produces results tables in a set of comma separated text files (*.csv).

4.1 Basic UnitCommit Results

These show up in …/models/ops/out/ after a successful model run [Use --out_dir=DIR to change the path relative to the model file. Default is out/]

- **UC_summary.csv:** Contains a detailed summary of the results (e.g. cost breakdown, emissions by type, etc.), configuration (e.g. input data files, option flag settings, etc.), and run information (e.g. solver time, number of equations, etc.) In many cases this will contain all of the output data you need. It is a strictly two column format with separate field names and values. The field names include units and should be self describing. Only alphanumeric symbols plus the underscore ("_") are used for fieldnames so they can readily be used as variable names, structure field ids, or dictionary/hash keys in an automated parsing program.
- **Other outputs:** Complete unit commitment plan, dispatch time series, and other files are also created in the same directory. [You can suppress printing these using the --summary_only=1 option] All of the output files also share the same prefix. [Defaults to "UC_" Can be changed using --out_prefix=NEW_PREFIX_].

4.2 Basic Features (UnitCommit)

The basic, no frills commitment setup uses:

- *Integer commitment decisions:* units are chosen to run or not for each time period using individual (binary) or clustered (0:n integer). Binary vs integer and the specific number of units available is automatically determined based on the total installed capacity (cap_cur) and the capacity per generation unit (gen_size [specified in either the test_gen.inc input file]. This makes for a simple Mixed-Integer problem (MIP). [Relax integer commitment constraints using --uc_lp=1.] This also enforces the minimum output constraints for units that are running;
- *No other "unit-commitment" constraints:* The basic call ignores most other constraints such as ramping [--ramp=1], startup costs [--startup=1], minimum up & down times [--min_up_down=1], etc.;

---

[2] These directions assume running from the command line. As described for StaticCapPlan, when running using the Windoze IDE, setup a new project in this same directory, open UnitCommit.gms and run as usual. In more complicated examples, the command line options can be entered into the command line option box near the top right of the IDE, just below the menu bar.

[3] Note: since the test set of data uses a non-sequential 20-block load duration curve, the commitment results with this basic call are rather arbitrary.

- *Allowed non-served energy:* When operational costs are too high, or constraints are too strict, load is shed for a price, specified by pPriceNonServed in the system definition file. [Use --no_nse=1 to disable non-served energy and force supply to equal demand, The cost of non-served energy is specified in the test_`sys.inc` file.]; and
- *Allowed wind/renewable shedding:* If/when cost effective, excess wind is shed (not used). [Use `--force_renewables=1` to disable shedding. Disabling shedding only is possible for lower quantities of renewables]

These basic features represent only a smalls subset of StaticCapPlan's features. Section 5 describes a more complex example and section 11 describes the complete set of configuration options.

5 A more advanced StaticCapPlan example

Extending the basic call with command line options enables much more sophisticated (and slower to run) capacity optimizations. For example[4]:

```
gams StaticCapPlan -errmsg=1 --unit_commit=1 --sys=gcnw_sys.inc
 --demand=ercot2007_dem_yr_as_14wk.inc --demscale=1.1
 --gparams=nw_power_plan6_gen_params.inc --derate=1
 --retire=0.4 --plan_margin=0.15 --rps=0.2 --min_gen_size=0.25
```

determines the least cost expansion plan:

- For a system with four types of generators: natural gas, coal, nuclear, and wind [from the `gcnw_gen.inc` file that is referenced by `gcnw_sys.inc`];
- Using 14 weeks of hourly time series data [because the use of `--demand=ercot2007_dem_yr_as_14wk.inc` overrides the default demand in `gcnw_sys.inc`]. These weeks of operation are scaled based on their weighted duration [dur field in `ercot2007_dem_yr_as_14wk.inc`] such that a full 365days of operating cost are computed;
- Scaled up for 10% load growth [`--demscale=1.1`];
- Assuming that 40% of the existing [e.g. that specified by `cap_cur` in `gcnw_gens.inc`] generating capacity has retired [`--retire=0.5`];
- Requiring 20% of the energy to come from renewable sources through an enforced renewable portfolio standard (RPS) [`--rps=0.2`];
- Requiring firm peak capacity [`cap_credit`] of 15% beyond the actual peak demand [`--plan_margin=0.15`, note that while this value overrides that specified in the sys file, but that the sys specified value is only used if this option is enabled by setting `plan_margin` to the special value of 1]. An important feature of the model is that it distinguishes between peak firm capacity [`cap_credit`] and average available (possibly derated) output. This captures the fact that renewables and thermal units with unreliable fuel sources may require additional "firming" capacity to ensure reliable peak period operation beyond that suggested based on average output level alone;
- Using cost and generator performance data from the Northwest Power Plan [`--gparams=nw_power_plan6_gen_params.inc` overrides the call by `gcnw_sys.inc` to the EIA cost data. Note that because `gcnw_gen.inc` only contains capacity data, all of the parameters (e.g. heat rate, gen_size, etc.) are taken from the gparams file. However, any fields with non-zero data in the gens file takes precedence over the gparams file.];
- That uses basic clustered unit commitment based operations (i.e. discrete on/off decisions with unit level minimum output constraints, but ignores other unit commitment constraints) [`--unit_commit=1`]; but
- Derates the generating capacity of all plants by their total outage rate to approximately capture maintenance and reserve requirements [`--derate=1`]; and
- Forces all generation investments in at least 250 MW chunks [`--min_gen_size=0.25`].

The GAMS option -errmsg=1, puts the helpful error code name/description inline with any errors in the *.lst file, rather than only listing the error numbers.

[4] Run from the command line in the …/models/capplan directory or using the command line option box near the top right of the Windoze IDE.

5.1 Other useful options

In addition to the options used in the example, all of the unit commitment options supported by UnitCommit can be used. Other, commonly useful planning include:

- All of the options supported by `UnitCommit`;
- `--ignore_integer=1`, to use continuous build (and unit commitment if enabled) decisions;
- `--max_solve_time=NUMBER`, to extend the optimization time out beyond the default of 3hrs. Times must be specified in seconds;
- `--from_scratch=1`, to ignore any existing capacity [`cap_cur`] and build the entire generator fleet from scratch;
- `--no_cap_limit=1`, to ignore any specified maximum capacities [cap_max] and enable arbitrary levels of expansion for all unit types;
- `--ignore_cap_credit=1,` to stop distinguishing between firm capacity credit and average derated availability when computing the planning margin;
- `--skip_cap_limit=1`, to disable a heuristic constraint that attempts to speed the optimization by limiting the sum of total capacity, in addition the capacity limits on a per unit-type basis. This may be necessary in some non-standard cases;
- `--derate_to_maint=1`, to derate power production only to that required by maintenance (planned outages), rather than the traditional derating to cover both planned and unplanned (forced) outages;
- `--plan_margin_penalty=NUMBER`, to relax the planning margin constraint, by providing an alternative mechanism to pay (as in a fine) for violating the planning margin;
- `--co2cap=NUMBER`, to enforce an annual CO2 (carbon) cap [in Mt-CO2 equivalent];
- `--force_gen_size=NUMBER`, to ignore the specified unit size [gen_size] and force a uniform generation unit capacity [in GW];
- `--calc_water=1`, to also compute generator and system water usage. This requires water data to be specified in the gens data file; and
- `--memo=STRING_WITH_NO_COMMAS`, to place the string, verbatim into the summary file. Useful for internally tagging runs by name, date, etc.

Additional details and a complete list of options is included in section 11.

6 A more advanced UnitCommit example

Like its capacity-planning sibling, the basic call to UnitCommit can be greatly extended using command line options. For example[5]:

```
gams StaticCapPlan -errmsg=1 --sys=ercot2007eGrid_sys.inc
 --demand=ercot2007_dem_yr_as_14wk.inc
 --min_gen_size=0.25 --gparams=eia_aeo2011_gen_params.inc
 --ramp=1 --startup=1 --min_up_down=1 --co2cost=75
 --rsrv=separate --no_nse=1 --maint=1
```

simulates least cost operations:

- For the ERCOT generating system from 2007 using units clustered by fuel and prime mover type [from the `ercot2007eGrid_units_clust.inc` generator file that is referenced by `ercot2007eGrid_sys.inc`];
- Using 14 weeks of hourly time series data [the use of `--demand=ercot2007_dem_yr_as_14wk.inc` overrides the default single week demand data file called in `ercot2007eGrid_sys.inc`];
- Using cost and generator performance data from the EIA[6] [`--gparams=neia_aeo2011_gen_params.inc` overrides the call by `ercot2007eGrid_sys.inc` to the Northwest Power Plan cost data. In this case, the `ercot2007eGrid_units_clust.inc` generator file specifies cluster average unit sizes and heat rates based on the actual existing ERCOT system so only cost and technical data are taken from the gparams file.];
- That uses a full clustered unit commitment optimization including:
    - Discrete on/off decisions with unit level minimum output constraints [default for Unit Commit, can disable with `--unit_commit=off`];
    - Hour-to-hour ramp rate limits [`--ramp=1`];
    - Startup costs [`--startup=1`];
    - Minimum up (running) and down (shutdown) constraints [`--min_up_down=1`]; and
    - Five separate classes of operating reserves [`--rsrv=separate`], specifically regulation up & down, spinning reserve/load following up, load following down, and off-line quick starting reserves. [The quantity of these reserves is specified by parameters in the `ercot2007eGrid_sys.inc` file and includes reserves as function of load and as a function of wind capacity and (forecast) production.]
- And simultaneously creates and uses an optimal week-by-week maintenance [`--maint=1`] plan that keeps units off-line long enough to meet their annual maintenance requirements [Since only 14 weeks of demand are used, the time period duration [dur specified in `ercot2007_dem_yr_as_14wk.inc`] is used to scale the maintenance periods such that being off-line for maintenance for one week, actually counts as 1 to 4.3 weeks of maintenance, depending on the week.];
- Is subject to a $75/ton-CO2 carbon tax/cost [`--co2cost=75`];

[5] Run from the command line in the …/models/ops directory or using the command line option box near the top right of the Windoze IDE.

[6] Note that the raw EIA data source, Annual Energy Outlook 2011, only specifies costs, unit sizes and heat rates. Additional unit-commitment technical parameters in the EIA gparams file are still based on the Northwest Power Plan assumptions.

- Does not allow any non-served energy [`--no_nse=1`];
- And groups the individual units within the cluster into at least 250 MW chunks or their gen_size parameter, which ever is larger [`--min_gen_size=0.25`].

The GAMS option `-errmsg=1` (with only a single dash), puts the helpful error code name/description inline with any errors in the *.lst file, rather than only listing the error numbers.

6.1 Other useful options

In addition to the options used in the example, all of the unit commitment options supported by UnitCommit can be used. Other, commonly useful operations options include:

- `--uc_lp=1`, to relax the integer constraints on commitment while still enforcing the desired unit commitment constraint equations (continuously) for accurate reserve computations;
- `--uc_int_unit_min=NUMBER`, to partially relax the integer commitment constraints for units less than the specified capacity [in GW];
- `--uc_ignore_unit_min=NUMBER,` to partially ignore commitment constraints all together and simply assume merit order dispatch for units less than the specified capacity [in GW];
- `--adj_rsrv_for_nse=1`, to allow the load-driven (but not renewable driven) reserve requirements to reduce when there is non-served energy. This allows non-served energy to provide a way of meeting reserve requirements for extreme cases at the cost of notably increased computation time. It is generally only needed for extreme cases with lots of non-served energy; and
- `--no_capital=1`, to completely ignore annualized capital costs, which otherwise are computed as part of the annual fixed costs.

Additional details and a complete list of options is included in section 11.

7 Data and Update Files

7.1 Introduction

In addition to command line options, data files and update files represent the key methods for customizing the model simulations.

The system **data** is divided among a number of files that can be mixed and matched as needed. By convention any new analysis typically creates a complete, unique set/copy of data files, typically with a common prefix, to prevent conflicts with future file updates. The test_* and ieee_rts96_* data files provide an example of this convention. The only exception is when analysis uses a commonly available family of demand & availability data in which case only custom sys, gen, and fuel files are required. The thesis_* data files demonstrate this usage. The g_param files are assumed to be universally useful and hence should be used without custom names (unless of course they need to be modified).

Update files are a useful way to manipulate data and model runs beyond what is possible with command line options alone, but without spawning a complete new set of data files. Update files are read-in after all of the data files are loaded, but before other command line options are applied. They are particularly useful for sensitivity analysis and automated calling of the models.

7.2 Input data files

These are all in …/models/data/ [Use --data_dir=DIR to change, path relative to model file, so default is ../data/]

- ***sys.inc:** Baseline system-wide configuration parameters such as WACC, planning reserves, etc. By convention, this file also calls ($include) the associated, standard set of additional data files. [Specified with --sys=FILE see Section 11 for more information]
- ***gens.inc (or *units.inc):** Contains high-level generator specific cost and technical data, including water numbers. This is the primary place to look/edit for different technology categories and mixes. [Change file name using --gens=FILE]. It is possible to put complete generator data into this file, but most of the time it is more efficient (and easier to maintain) only the generator specific parameters and use the "`gen_params`" file contains the lower-level unit commitment constraints: ramp rates, min output percentages, etc. The way this works is that any data not explicitly specified in the unit file (or listed as zero), is taken instead from the `gen_params` file based on the unit type code.
- **`*avail.inc`**: Time varying wind (and other time varying generator availability) output is specified in the "avail" file. Generation availability/renewable output. This is place for renewable energy production time series data. It can also be used for pre-defined maintenance schedules, co-generation, seasonal plants, and other time varying availability. Availability time periods are matched to demand time periods in one of two ways, either:
    - For simple data, both demand and avail have identical time block (B) and time sub-period (T) sets, with matching names and are simply matched one-to-one [see for example test_dem_20blk.inc and test_avail_20blk.inc]; or
    - To share availability data across multiple demand data files, it is possible to specify the availability data in a D_AVAIL set and then in the demand data file, an avail_idx field is used to refer to the corresponding availability data by line number (internally based on GAMS's ord() function) [for example see ercot2007_avail.inc

and the ercot2007_dem _*.inc family of demand files]. In this setup, the availability file typically covers the entire 8760hour year. The demand file ("dem") on the other hand specifies the hourly demand for the desired model time frame. My model automatically matches up the corresponding subset of demand blocks by name, but you might need to do this manually. For instance the week I am looking at begins at t1800, which will be the 1800th entry in the availability data.
    - IMPORTANT: Availability must be defined for all generator types, typically using a single line to define all time periods for thermal generators, with a separate table of hour-by-hour (or other time sub-period) for renewables. Use caution to exclude the renewables from the single line definitions or the data may not updated as expected. [Change file name using --avail=FILE see Section 11 for more information]
- ***demand.inc:** Contains the demand timeseries. [Use --demscale=1.2 to linearly scale up for 20% growth, or --demand=NEWFILE to specify an alternate file. Section 11 contains additional details]
- ***fuel.inc:** contains fuel prices and CO2 intensity (In this run the CO2 price is zero so just tracked for accounting). This data is used by the model to build up the total variable cost from the variable O&M cost plus the fuel cost using the heatrate. Similarly the startup costs have a fixed cost plus a fuel cost based component. [Change file name using --fuel=FILE]

7.3 Data Sets

A number of ready-to-go datasets are included in the …/models/data directory. The most highly developed (and hence best tested) data sets include:

- The `ercot2007*` family which is based on the actual demand and wind time series for the Electric Reliability Council of Texas (ERCOT) in 2007. Generator data is taken from eGrid with technical data from other sources;
- The `ieee_rts96*` family with complete data from the IEEE Reliability Test System (1996 version);
- The `test*` family which provides a constant set of default test data.

7.4 Update Files

Update files provide an easy way to update specific values and settings. They are pure GAMS code that is simply merged in with $include after loading all of the data files. As a result, settings in the update file override those from the data files. But, explicit command-line options are processed last and hence will override (or scale) parameters setup in the update file.

When creating update files it is important to note that although the core input data files will automatically create a single scenario to fill the scenario (S) set. However, the update file is applied after the introduction of S (scenario) space. As a result, most parameters must be indexed by S and you must use the scenario dependent parameters: pFuel, pDemand, pGen, and pGenAvail. Rather than the raw (scenario independent) data parameters (pGenData, pDemandData, etc). Changes made to the to the p*Data parameters in the update file will NOT be used.
Look at …/models/data/example_update.inc for an example.

8 Stochastic Optimization (StocCapPlan and StocUC)

Although the core models, StaticCapPlan and UnitCommit are designed for static, deterministic analysis, the use of the additional set "S" makes it very easy to call the basic models as part of a simple, deterministic equivalent stochastic programming problem. The StocCapPlan and StocUC models demonstrate how this can be done. Each one adds a few very simple constraints across the scenarios (e.g. installed capacity must be the same). A user provided scenario file then defines which parameters vary across scenarios.

In addition, the MATLAB folder includes an alternative approach to stochastic optimization using Dynamic Programming (DP, via backward induction) or Approximate Dynamic Programming (ADP, via double pass algorithm). These approaches allow more flexibility and powerful analysis at the expense of increased overhead, and potentially increased run time due partially to the need to pass information between MATLAB and GAMS.

9 Additional Tools

9.1 Data helpers

These excel files include the raw data and intermediate calculations to create most of the data files described above. In most cases you can use these for simpler formatting and then copy and paste into the text files above. These are located in the …/raw_data directory

IMPORTANT: If you make changes to these files, particularly using new data, please save them using a new name. For all other files, please do not use a new name.

- **ERCOT_2007_Hourly_Wind&Load_GAMS_formatter.xlsx:** Contains the raw 2007 wind and load time series and a number of nifty page formatters to create block load duration curves, selected weeks, and 8760 series sorted by time, load, etc. All such pages are designed for easy copy and paste into GAMS data files. T
- **ERCOT2007 Unit Summary.xlsx:** All the pieces I used to go from the raw eGrid data to a filtered list of generators and from there on to various clusters. The filters and cluster settings are dynamic and the gen list table is separate to allow for easy updates.
- **Generator Data Formatter.xlsx:** (located in …/models/data) Provides a table based interface for editing generator parameter include data. Includes full source data and in-line notes on all assumptions.

9.2 MATLAB tools

In addition to the work described here for the StaticCapPlan and UnitCommit models, I developed an extensive collection of MATLAB functions to interact with these GAMS models. These files use the same data files (from …/models/data) as the GAMS models. The complete code implements DP and ADP wrappers around the GAMS models for use in stochastic multi-period analysis. Many of these tools are also useful on their own. Specifically (all located in …/models/MATLAB):

- **`Dispatch.m`**: a generically useful MATLAB dispatch operations model with CO2 cost support and a rich set of data output.
- **`wind2power.m`**: wind speed to power production function based on high quality interpolation of actual Enercon turbine. Also includes the industry standard height adjustment calculations to scale measured data up to planned hub height.
- **`CapPlanDpCallGams.m, CapPlanDpWriteGamsUpdate.m, CapPlanDpParseSummary.m`**: A set of generically useful files for setting up, calling, and post-processing UnitCommit, StaticCapPlan and related models within MATLAB.
- **`CpDpReadGenData.m`**: Read in UnitCommit/StaticCapPlan model generator data files and merge in g_param files for missing fields.
- **`testCapPlanADP.m, timeCapPlanADP.m`**: Two simple examples of how to use the DP and ADP code. These run through a complete analysis of two different simple test problems.
- **`CpDpScenario.m`**: An object oriented class definition to enable standard setup, running, and processing of DP/ADP scenarios. Extensive use of inheritance greatly reduce the amount of duplicate code when using new scenarios. Examples of derivative scenarios include: `scenTest.m, scen2x2.m, scenGCNW.m, and scenGCW.n`.
- **`OpsDiff.m, OpsRead.m, OpsSummary.m`**: A set of files designed for reading and comparing output data from `UnitCommit` and related models. Used extensively in my dissertation.

- **`PlanDiff.m`**: A similar file for comparing the results of `StaticCapPlan` and related models. Used extensively in my dissertation.
- **`CapPlanDpInit.m`**: Sets up the large data structure used by the DP and ADP code. Key to understanding how most of the other `CapPlanDp*` functions work.

Important notes for these MATLAB files:

- In general these MATLAB files are ready to use and have help text comments compatible with the MATLAB help system to get you started;
- Much of this code requires functions from the ADP toolbox that I also wrote. It should be installed in a separate directory and added to the path in MATLAB.
- Double check the code history table below the help text in each file. At the time of this writing, most recent active MATLAB development was in early/mid-2012. If there hasn't been more recent activity it might be an inactive piece of code.
- Many of the GAMS interaction tools assume that the environment variable "GAMS" is set to the full path of the GAMS executable. On OSX, add something like `PATH="${PATH}:/Applications/GAMS"` to your `.profile`. Under windows you have to use System | Advanced Settings in the Control Panel.
- Take a look at the "scratchpad" files for example usage.

9.3 Additional Stand alone models and tools

- **`…/models/capplan/Screening Curves.xlsx`:** A simple graphical capacity planning model that includes support for carbon costs and renewable capacity credit. Makes nice pictures if you need them, and provides a foundation for understanding cost trade-offs.
- **`…/models/util/ldc.py`**: A nifty command line program (in python) for plotting pretty load duration curves, weekly time series, and more.
- **`…/models/util/put2csv.gms`**: A very handy utility for creating comma separated value output tables from GAMS. This code is easy to use more powerful and more flexible than Rutherford's gams2csv utility. The example code in **`csv_read_write_demo.gms`** demonstrates how the utility works.

10 Directory Structure

10.1 …/docs

Location of this and other documentation files. Also extensive help and documentation is found embedded in all code files and most spreadsheets.

10.2 …/models

The models sub-directory contains all of the model and data files for both GAMS models (StaticCapPlan and UnitCommit). In addition it contains MATLAB code and many other tools.

10.2.1 …/models/data

Contains the complete set of data files used in my thesis and a number of other analyses. See section 7 for more information.

10.2.2 …/models/capplan

Contains the core `StaticCapPlan.gms`, `StocCapPlan.gms` along with other capacity planning models.

10.2.3 …/models/ops

Contains `UnitCommit.gms` and other less developed operations-only models.

10.2.4 …/models/shared

Contains a full set of shared "functions" used by the core models. This includes a GAMS sub-models for computing most of the more advanced model features such as minimum up and down times, planning margins, reserves, etc. In addition:

- **…/models/shared/writeSummary** and **calcSummary:** provide most of the calculations and output for the summary file. Look here to find out the source for the summary output file

10.2.5 …/models/util

Contains a large number of generically useful utilities including the GAMS output formatter gams2csv which is more flexible than other similar tools, a load duration curve plotter, and a number of Excel files.

10.2.6 …/models/config

Contains the shell scripts I used when submitting jobs to the Svante cluster's queue system. Includes examples of best practices for copying models to the compute node's scratch drive to minimize intra-cluster network bandwidth for GAMS temp files.

10.2.7 …/models/MATLAB

Contains a large number of tools for interacting with these models through MATLAB. Also contains a mostly operational Approximate Dynamic Programming wrapper for these models. See section 9.3 for more details.

10.3 …/raw_data

Yep, you guessed it, a set of (fairly) raw data that could be generically useful. This includes a number of helpful Excel files for generating the data files needed by the GAMS models (and hence

stored in …/models/data). In addition I do have lots more data in even less refined form that is not included in this directory. Let me know if you are interested.

10.4 …/results

Here is where all of my results (to date, at time of release) are stored. These are divided into data folders based on the type of analysis.

Many of these results are text file heavy, and hence stored as ZIP fies to save space. Even compressed this is still a large directory, and hence not included in the model only distributions.

11 Complete model options

By default, StaticCapPlan and UnitCommit use a very simplistic models. Additional features are enabled by command-line options. A complete upt-to-date list of command-line options is included in the text at the top of the model files themselves, StaticCapPlan.gms and UnitCommit.gms, respectively. In addition helper files (in the …/models/shared directory) include a list of these control variables that are implemented in the associated file (and any sub-files if needed). The table below lists the complete set of options available at the time this document was produced. However, since model extensions are expected to add model flags rather than comment code in/out, this list may continue to expand.

Option with default	StaticCapPlan	UnitCommit	Description
Data Related. These specify the input files. Files are assumed to come from ../data relative to the model directory [directory can be changed with --data_dir]			
--sys=test_sys.inc	X	X	System parameters include file. This file references all data for a model run. Typically single value data such as: cost of carbon, WACC, etc. are included directly, while larger tables are in separate sub-include files[7]. The standard sub-include files are: • fuel.inc Fuel names, prices, and emissions • gens.inc Generator list, costs, and operating parameters. • demand.inc Demand block set, duration, and power levels
--demand=(from sys)	X	X	Demand include file that defines demand blocks, levels, and duration. Overrides the demand sub-include file from sys.
--fuel=(from sys)	X	X	Fuel prices and emissions. Overrides the fuel sub-include file from sys.
--gens=(from sys)	X	X	Generation set & tables of parameters & availability/renewable output. Incomplete data is taken from the optional gparams file if specified (otherwise missing data is treated as zero). Overrides the gen sub-include file from sys.
--gparams=(OPTIONAL from sys)	X	X	Default generator parameters to use for any missing or zero values from the generator data file. Parameters are matched to the generators based on the type field. This allows only

[7] See --update and the next section on specific value overrides for alternative ways of specifying or adjusting these values

			specifying only a subset of generator parameters in the gens file (e.g. name, type, cap_cur, and cap_max) and then extracting remaining cost and technical parameters from a common dataset.
--avail=(from sys)	X	X	Generation availability/renewable output. This is place for renewable energy production time series data. It can also be used for pre-defined maintenance schedules, co-generation, seasonal plants, and other time varying availability. Availability time periods are matched to demand time periods in one of two ways, either: • For simple data, both demand and avail have identical time block (B) and time sub-period (T) sets, with matching names and are simply matched one-to-one [see for example test_dem_20blk.inc and test_avail_20blk.inc]; or • To share availability data across multiple demand data files, it is possible to specify the availability data in a D_AVAIL set and then in the demand data file, an avail_idx field is used to refer to the corresponding availability data by line number (internally based on GAMS's ord() function) [for example see ercot2007_avail.inc and the ercot2007_dem _*.inc family of demand files] IMPORTANT: Availability must be defined for all generator types, typically using a single line to define all time periods for thermal generators, with a separate table of hour-by-hour (or other time sub-period) for renewables. Use caution to exclude the renewables from the single line definitions or the data may not updated as expected.
--update=NONE	X	X	[optional] After loading the data files, load the specified update file from the .../models/capplan directory. Settings in the update file override those from the data files. Does not override any explicit command-line options. Best practice for long-lived update files (as opposed to computer generated and then automatically deleted) is to place them in …/models/data and use update=../data/changes.inc Look at example_update.inc for an example. IMPORTANT: the update file works in S (scenario) space, so most parameters must be indexed by S and you must use the scenario dependent parameters: pFuel, pDemand, pGen, and pGenAvail. Changes to the p*Data parameters (pGenData, pDemandData, etc) will NOT be used.
--scen=NONE	X	X	For multiple scenario problems (multi-period or stochastic) specifies the list of scenarios (populates the S set) and their associated weight/probability table, pScenWeight(S).

Specific Value Overrides These take precedence over all values defined in data files and are useful for sensitivity analysis, etc. IMPORTANT, these values are used for ALL scenarios, use a scenario or update file for changing these on a by scenario basis.			
--co2cost=#	X	X	Cost of CO2 in $/t-co2e (default: use sys or update value)
--demscale=#	X	X	Factor by which to uniformly scale demand. A value of 1 implies no scaling. (default: use sys or update value)
--rps=#	X	X	Renewable Portfolio Standard in the range 0 to 1 (default: use sys or update value)
--co2cap=#	X	X	Carbon Emission Cap (Mt-co2e) (default: use sys or update value)
Model Setup Flags By default these are not set. Unless otherwise specified they can't be toggled but rather will be enabled when set to *any* value, including 0, off, or zero.			
--obj_var=vTotalCost **--obj_var=vOpsCost**	X	X	Variable to minimize in solution. In scenario mode (stochastic UC, multi-period planning, etc.) The weighted sum across scenarios of this value is used by default. Common options include: • vTotalCost (default for StaticCapPlan) Minimize total costs; • vOPpaCost (default for UnitCommit) Minimize operations costs; or • vCarbonEmissions Use with no_nse=1 to find minimum possible co2 Technical Note: Any variable indexed by S can be used
--startup=(off)	X	X	Compute startup costs (also enables unit_commit) (default: ignore)
--unit_commit=(off)	X	X	Compute unit commitment constraints (default: ignore)
--ramp=(off)	X	X	Flag to limit inter period ramp rates (default: ignore)
--ignore_integer=(off)	X	X	Flag to ignore integer constraints in new capacity investments, (eg allow 1MW nuclear plants) and in Unit Commitment if enabled (unit is either committed or not) (default: use integer constraints)
--avg_avail=(off)	X	X	Flag to use the average rather than time dependent availabilities. Using averages is OK for thermal units, but highly simplifies time varying renewables. This simplification is made in the analytic version of the model, but not generally a good idea for numeric estimates. (default: use complete time varying information.)
--ignore_cap_credit=(off)	X	X	Flag to ignore the normal distinction between capacity credit and availability. When set, the capacity credit parameter is set equal to the time weighted average of availability. (default: use cap_credit value from GenParams)

--uc_ignore_unit_min=(0)	X	X	Threshold for unit_min to ignore integer commitment decisions in unit commitment. Gens with unit_min less than or equal to this value will be treated as continuous to speed performance
--uc_int_unit_min=(0)	X	X	Threshold for unit_min to ignore INTEGER commitment decisions & constraints. Gens with unit_min less than or equal to this value will still have commitment variables, but their valid range is relaxed to be continous. The same equations are used as for those units with integer constraints.
--uc_lp=(0)	X	X	Ignore integer constraints on all UC variables (& startup/shutdown)
--adj_rsrv_for_nse=(off)	X	X	Adjust reserves for non-served energy. This uses actual power production rather than total desired demand for setting reserve requirements. This distinction is only significant if there is non-served energy. When enabled, non-served energy provides a way to reduce reserve requirements. [Default= use total non-adjusted demand]
--rsrv=(none)	X	X	Specify Type of reserve calculation. Options are: =separate Enforce separate reserve requirements based on "classic" ancillary services plus additions for renewable uncertainty. This includes Reg Up, Reg Down, Spin Up, & Quick Start =flex Use combined "flexibility" reserves grouped simply into flex up (Spinning + QuickStart + RegUp + Renewable Up) and flex down (RegDown + Renewable Down). =both Compute both separate and flexibility reserves =(none) If not set, no reserve limits are computed
--non_uc_rsrv_up_offline=0	X	X	For *non-unit commitmen*t generators, the fraction of non-running generation capacity to use toward UP reserves. This parameter has no effect on UC generators. deJonge assumes 0.6, NETPLAN assumes 1.0, (default=0).
--non_uc_rsrv_down_offline=0	X	X	For *non-unit commitment generators*, the fraction of non-running generation capacity to use toward DOWN reserves. This parameter has no effect on UC generators. deJonge assumes 0.6, NETPLAN assumes 1.0, (default=0).
--no_quick_st=(off)	X	X	Flag to zero out quickstart reserve contribution to spinning/flex up reserves. Useful when non_uc_rsrv... > 0
--no_nse=(off)	X	X	Don't allow non-served energy
--force_renewables=(off)	X	X	Force all renewable output to be used. This is only feasible until the point where load and op_reserves dictate a max. (until we add storage). When used with cap_fix, it is a bit more widely useful b/c we can limit output to the level of demand. (this is NLP when capacity is a decision)
--fix_cap=(off)	X		Fix capacity to cap_cur by not allowing additions or

			retirements. i.e. Do not make plant investment decisions. Instead only compute operations for the current plant capacities
--max_start=(off)	X	X	Enforce maximum number of startups (default: ignore)
--force_gen_size=(off)	X	X	Force all plant sizes to equal the specified value (in GW)
--min_gen_size=(off)	X	X	Force small plant sizes to be greater than or equal to the specified value (in GW)
--derate=(off)	X	X	Enable simple derating of capacity to approximate outages, reserves, and maintenance. This is standard practice for simple capacity planning models. *Important Note: The maximum power available (for power and reserves) for any generator during any period is given by the **minimum** of the derate constant and time varying availability. Often thermal plant availability has only been reduced by the unplanned (forced) outage rate, which is appropriate for use with more sophisticated operating reserves. When these reserves are not used, the more conservative derate values, which commonly are include both planned (e.g. maintenance) and unplanned outages are recommended.*
--from_scratch=(off)	X		Ignore the current capacity specified in the generator data file and re-build the generation mix from scratch. *WARNING: Be sure to set non-zero costs for all generator types when using this option. For example, in the main ERCOT data (ERCOT2007), the capital costs for existing technologies are set to zero, which will result in a stupid mix from scratch.*
--no_cap_limit=(off)	X		Allow unlimited expansion of all generators (useful with from_scratch)
--basic_pmin=(off)	X	X	Forces generators to always output at least p_min. It enforces a permanent minimum output level for generating units with non-zero p_min values. This forces the generation to always be on. This can be useful for baseload plants with simple (non-UC) operations.
--no_capital=(off)	X	X	Ignore capital costs, used for operations models to only compute non-capital costs. Not recommended for planning models. [default: include capital costs]
--renew_lim=(avg)	X		Technique for limiting renewable expansion. Valid options: =avg Limit average output to demand peak: average power < peak*(1+renew_overbuild) [default] =firm Limit based on firm capacity (typically way to high): cap_credit < peak*(1+plan_margin)*(1+renew_overbuild) =rps Limit expansion to that required to meet the RPS (maybe too low for high rps):

			rps*(1+renew_overbuild) =norm Treat As Normal Gen (uses general overbuild, not renew_specific): (1+overbuild)*max(avg < peak*(1+plan), cap_credit < peak(1+plan))
--overbuild=0.2	X		Amount (a fraction) over the planning margin to limit the maximum number of plants for each type. Also used with the heuristic capacity limit described below.
--renew_overbuild=0.2	X		Amount (a fraction) over the peak/rps energy requirements to consider overbuilding for renewables. May have to be raised with very high RPS levels.
--skip_cap_limit=(off)	X		[Recommended when computation time allows & used with all thesis model runs] Do not enforce the heuristic capacity limit equation that can greatly speed MIP tree searches by ignoring capacity combinations, such as maxing out all gens, that exceed the tougher of the planning margin or operating reserve requirements by more than the overbuild factor. In rare cases, with few generator types, strange availability patterns, etc. this heuristic may be overly restrictive.
--no_loop=(off)	X	X	Do not loop around demand periods for inter-period constraints such as ramping, min_up_down. (default= assume looping)
--maint=(off)	X	X	Compute Maintenance schedule (default = use avail data, typically assumes full availability for thermal plants)
--maint_lp=(off)	X	X	Relax integer constraints on maintenance decisions (default: use integers)
--maint_om_fract=0.5	X	X	If maintenance planning enabled, the default fraction of total fixed O&M costs to divide among the required weeks of maintenance.
--plan_margin=(off)	X	X	Enforce the planning margin. Set to 1 to enable and use the problem defined pPlanReserve (typically in sys.inc). Alternatively can set to a value < 1 that then is used for pPlanReserve overriding other definitions. This ensures adequate supply at peak under simplistic uncertainties, by requiring sufficient firm capacity (based on cap_credit) to meet the peak demand plus the margin specifed by pPlanReserve in the system data file. This is standard practice for simple capacity planning models.
--plan_margin_penalty=(off)	X	X	Allow planning margin to be not met and define associated penalty [$/MW-firm] (default= must meet planning margin)
--rps_penalty=(off)	X	X	Allow planning margin to be not met and define associated penalty [$/MWh] (default= must meet rps)
--retire=(0)	X	X	Fraction of current capacity to retire. Max capacity is also adjusted down accordingly (value 0 to 1)

--derate_to_maint=(off)	X	X	Override gen datafile derating value and derate based on the maintenance value only.
Solver Options			
--max_solve_time=10800	X	X	Maximum number of seconds to let the solver run. (Default = 3hrs)
--mip_gap=0.001	X	X	max MIP gap to treat as valid solution
--par_threads=1	X	X	Number of parallel threads to use. Specify 0 to use one thread per core (Default = use only 1 thread)
--par_mode=1	X	X	CPLEX parallel mode 1=deterministic & repeatable, 0=automatic, -1=Opportunistic, but not repeatable (Default = determinstic)
--lp_method=4	X	X	CPLEX code for lp_method to use for pure root node, LP, RMIP, and final MIP solve. Options: 0=automatic, 2=Dual Simplex, 4=barrier, 6=concurrent (a race between dual simplex and barrier in parallel) (Default = 4, barrier) Use 6 if running in parallel
--cheat=(off)	X	X	use epsilon-optimal branch & bound by removing solutions that are not "cheat" better than the current best. This can speed up the MIP search, but may miss the true optimal solution. Note that this value is specified in absolute terms of the objective function.
--rel_cheat=(off)	X	X	Similar to cheat, but specified in relative percentage of objective this works in CPLEX only
--probe=0	X	X	CPLEX code for probing, a technique to more fully examine a MIP problem before starting branch-and-cut. Can sometimes dramatically reduce run times. Options: 0=automatic, 1=limited, 2=more, 3=full, -1=off. (default = 0, automatic). Probe time also limited to 5min.
--priority=off	X	X	Use branching priorities for Branch & Bound tree, set to anything other than off to enable.
Output Control Flags **by default these are not set. Set to any number, including zero, to enable**			
-errmsg=(off)	X	X	A **GAMS** specified option, hence the need for only a single dash, to put descriptive text for error messages at error sites within the *.lst file. Without this option, error descriptions are unhelpfully located toward the end of the *.lst file and only the numeric code is listed at the site of the error.
--memo=(none)	X	X	Add this text to the summary output file. DO NOT INCLUDE COMMAS (Default: none)
--debug=(off)	X	X	Print out extra material in the *.lst file (timing, variables, equations, etc)
--debug_avail=(off)	X	X	Display full availability table in *.lst file for debugging
--debug_off_maint=(off)	X	X	Create table of capacity off maintenance

--no_csv=(off)	X	X	Flag to suppress creation of csv output files (default: create csv output)
--summary_only=(off)	X	X	Only create output summary data (default: create additional tables)
--summary_and_power_only=(off)	X	X	Create only summary & power table outputs (Default: all files)
--out_gen_params=(off)	X	X	Create output file listing generator parameter input data (Default: skip)
--out_gen_avail=(off)	X	X	Create output file listing generator availability input data (Default: skip)
--gdx=(off)	X	X	Export the entire solved model to a gdx file in the out_dir (Default: no gdx file)
--data_dir=../data/	X	X	Relative path to data file includes
--out_dir=out/	X	X	Relative path to output csv files
--util_dir=../util/	X	X	Relative path to csv printing and other utilities
--shared_dir=../shared/	X	X	Relative path to shared model components
--out_prefix=[varies]	X	X	Prefix for all output data files. Defaults vary with each model. For example SCP_ for StaticCapPlan and UC_ for UnitCommit
Additional Water Use/Limit Calculation Options These require additional water related data fields in the gen (or gparam) data files. Specifically the variable water withdrawal[8], h2o_withdraw_var, (gal/MMBTU) and optionally a generation level water limit, h2o_withdraw_max, (Tgal) are used.			
--calc_water=(off)	X	X	Enables the calculation of water usage (withdrawl), costs, and limits. Only works if the active generator data file contains water usage information. If not you will get errors
--h2o_limit=(Inf)	X	X	System wide maximum water use [Tgal]. Only computed for gens with specified water usage (h2o_withdraw_var)
--h2o_cost=(0)	X	X	System wide water cost [$/kgal]. Only applied to gens with specified water usage (h2o_withdraw_var)

[8] Note that there can be a significant difference between total water withdrawal--the amount taken out of the water source--and water consumption--the amount of water that evaporates or is otherwise no longer usable. This is because certain cooling schemes, such as once-through-cooling, may extract significant amounts of water, but then return the majority back to the water source. In this version of the model, only a single type of water use is handled. Although the text refers to withdrawals, the user could instead use consumption by adjusting the variable water use numbers accordingly.

12 Customizing

12.1 Intro, Design Philosophy & Request

I put a lot of effort into making these models modular and highly reusable. In particular, if you are adding and changing things, please, please, do so with the same philosophy. For example rather than commenting out a line or ten, make a new data file or add a command line option. There are no changes (except bug fixes) required to the core model to re-run any of the analyses I've done through out the 4 years of my thesis. Instead as things were added or needs changed, I simply added a new command line option so that all of the old code would still run.

Likewise for the MATLAB code, options are NOT hard coded, but rather passed as parameters (often through an option stucture). This makes the code much more reusable and can greatly simplify debugging, expansion, and version control.

If you develop more broadly useful data sets, new model features, or more, please share them with me so they can be included in the next release. If you are doing extensive development I am happy to set you up as a contributor in the SVN repository.

12.2 Data Files

Go for it. This is the place to make the most of the customizations. Best practice is to make a new copy with a new name for the complete set of data files, or at least the core ones (i.e. sys, gens, fuel) you might change, such that it is easy to know which files go together.

12.3 New Model Features

The models are intended to extended. Please follow the existing file of adding the bulk of the changes in one more additional files like those in the …/models/shared directory. Make new command line options (aka control variables in GAMS-speak). For a concrete example check out how the water computations are done. There are only a few simple include lines in the main code and the bulk of the work is done in the separate include files (all in …/models/shared): WaterEquations.gms, WaterDataSetup.gms, and writeWaterSummary.gms

12.4 Developer Notes

- To enable all this flexibility, the GAMS code makes extensive use of control variables, which can be defined at the command line using --NAME and accessed in the code as %NAME%.
- There are also a few macros
- The code makes extensive use of $include, $if, and $ifthen and related constructs. These are pre-processor directives meaning that GAMS first runs through the entire code opening additional files an including and excluding portions of the text to create a giant (temporary) file that is actually then passed to GAMS.
- The McCarl users guide (aka bigdocs in the GAMS help folders) is very useful and describes many features not easily found in the normal users guide. Download it from the GAMS site.
- I make extensive use of subversion (SVN) version control during development. Let me know if you are doing extensive development, and I can set you up as a contributor.