

Welcome to the Regional Energy Deployment System (ReEDS) Model!

This GitHub repository contains the source code for NREL’s ReEDS model. Users of this source code agree to the ReEDS licensing agreement <https://nrel.gov/analysis/reeds/request-access.html>. The ReEDS Version 2019 source code is available at no cost from the National Renewable Energy Laboratory. The ReEDS model can be downloaded or cloned from <https://github.com/NREL/ReEDS_OpenAccess>. New users must request access to the ReEDS repository through <https://nrel.gov/analysis/reeds/request-access.html>.

Contents

[Introduction (https://www.nrel.gov/analysis/reeds/) 2](#_Toc20758157)

[Required Software 3](#_Toc20758158)

[Setting up your computer to run ReEDS for the first time (for Microsoft Windows 10) 3](#_Toc20758159)

[ReEDS Repository Configuration 5](#_Toc20758160)

[Python Configuration 6](#_Toc20758161)

[GAMS Configuration 8](#_Toc20758162)

[R Configuration 9](#_Toc20758163)

[Executing the Model 10](#_Toc20758164)

[Prompts for user input during “runbatch.py” 15](#_Toc20758165)

[Runbatch.py Execution Protocol: 17](#_Toc20758166)

[Case-specific Batch File Execution Protocol 18](#_Toc20758167)

[Documentation 19](#_Toc20758168)

[Model Architecture 22](#_Toc20758169)

[Modules 22](#_Toc20758170)

[Tracking Capital Stock 24](#_Toc20758171)

[Frequently Asked Questions 25](#_Toc20758172)

[How much are the GAMS licensing fees? 25](#_Toc20758173)

[Is there a trial version of the GAMS license so I can test ReEDS? 26](#_Toc20758174)

[What computer hardware is necessary to run ReEDS? 26](#_Toc20758175)

[Contact Us: 27](#_Toc20758176)

[Appendix: 27](#_Toc20758177)

[ReEDS Model Switches (specified by user in “cases.csv”) 27](#_Toc20758178)

[Modeling horizon switches 27](#_Toc20758179)

[Demand switches 27](#_Toc20758180)

[Renewable energy variability switches 28](#_Toc20758181)

[Technology cost and performance switches 28](#_Toc20758182)

[Region switches 30](#_Toc20758183)

[Capacity prescription, retirement, and refurbishment switches 32](#_Toc20758184)

[Capacity growth limit switches 32](#_Toc20758185)

[Policy switches 33](#_Toc20758186)

[Technology inclusion switches 34](#_Toc20758187)

[Other model constraint switches 35](#_Toc20758188)

[Linear programming switches 35](#_Toc20758189)

===========================================================================

# Introduction (<https://www.nrel.gov/analysis/reeds/>)

The Regional Energy Deployment System (ReEDS) is a capacity planning and dispatch model for the North American electricity system.

As NREL's flagship long-term power sector model, ReEDS has served as the primary analytic tool for many studies (<https://www.nrel.gov/analysis/reeds/publications.html>) of important energy sector research questions, including clean energy policy, renewable grid integration, technology innovation, and forward-looking issues of the generation and transmission infrastructure. Data from the most recent base case and a suite of Standard Scenarios are provided.

ReEDS uses high spatial resolution and high-fidelity modeling. Though it covers a broad geographic and technological scope, ReEDS is designed to reflect the regional attributes of energy production and consumption. Unique among long-term capacity expansion models, ReEDS possesses advanced algorithms and data to represent the cost and value of variable renewable energy; the full suite of other major generation technologies, including fossil and nuclear; and transmission and storage expansion options. Used in combination with other NREL tools, data, and expertise, ReEDS can provide objective and comprehensive electricity system futures.

===========================================================================

# Required Software

The ReEDS model is written primarily in GAMS with auxiliary modules written in Python and R. At present, NREL uses the following software versions: GAMS 24.7.4; Python 3.6.5; R 3.4.4;. Other versions of these software may be compatible with ReEDS, but NREL has not tested other versions at this time.

GAMS is a mathematical programming software from the GAMS Development Corporation. “The use of GAMS beyond the limits of the free demo system requires the presence of a valid GAMS license file.”[[1]](#footnote-1) The ReEDS model requires the GAMS Base Module and a linear programming (LP) solver (e.g., CPLEX). The LP solver should be connected to GAMS with either a GAMS/Solver license or a GAMS/Solver-Link license. “A GAMS/Solver connects the GAMS Base module to a particular solver and includes a license for this solver to be used through GAMS. It is not necessary to install additional software. A GAMS/Solver-Link connects the GAMS Base Module to a particular solver, but does not include a license for the solver. It may be necessary to install additional software before the solver can be used.”[[2]](#footnote-2)

NREL subscribes to the GAMS/CPLEX license for the LP solver, but open-source and free, internet-based services are also available. The [*COIN-OR Optimization Suite*](https://www.coin-or.org/downloading/) includes open-source solvers that can be linked with GAMS through the GAMS Base Module. NREL has not tested the performance of these open-source solvers for ReEDS. The [*NEOS Server*](https://neos-server.org/neos/) is afree, internet-based service for solving numerical optimization problems*.* Links with NEOS can be made through [*KESTREL*](https://www.gams.com/latest/docs/S_KESTREL.html) which is included in GAMS Base Module. In its current form, ReEDS cannot be solved using NEOS due to the 16 MB limit on submissions to the server. However, modifications *could* be made to ReEDS to *potentially* reduce the data below to the required submission size.

Python is “an object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.” [[3]](#footnote-3) “Python is developed under an OSI-approved open source license, making it freely usable and distributable, even for commercial use. Python's license is administered by the Python Software Foundation.”[[4]](#footnote-4) NREL uses Conda to build the python environment necessary for ReEDS. Conda is a “package, dependency and environment management for any language.”[[5]](#footnote-5)

“R is a language and environment for statistical computing and graphics…R is available as Free Software under the terms of the Free Software Foundation’s GNU General Public License in source code form.”[[6]](#footnote-6)

===========================================================================

# Setting up your computer to run ReEDS for the first time (for Microsoft Windows 10)

The setup and execution of the ReEDS model can be accomplished using a command-line interpreter application and launching a command line interface (referred to as a “terminal window” in this document). For example, initiating the Windows Command Prompt application, i.e., cmd.exe, will launch a terminal window (Figure 1).

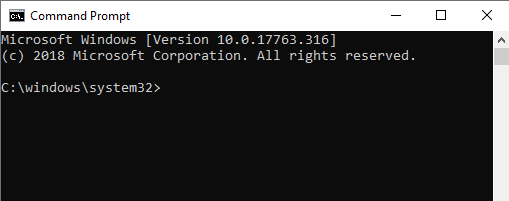


Figure 1. Screenshot of a Windows Command Prompt terminal window.

**SUGGESTON:** use a command line emulator such as ConEmu (<https://conemu.github.io/>) for a more user-friendly terminal. The screenshots of terminal windows shown in this document taken using ConEmu.

**IMPORTANT:** Users should exercise Administrative Privileges when installing software. For example, right click on the installer executable for one of the required software (e.g., Anaconda3-2019.07-Windows-x86\_64.exe) and click on “Run as administrator” (Figure 2). Alternatively, right click on the executable for the command line interface (e.g., Command Prompt) and click on “Run as administrator” (Figure 3). Then run the required software installer executables from the command line.

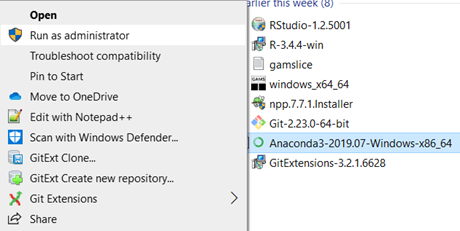


Figure . Screenshot of running an installer executable using “Run as administrator”.

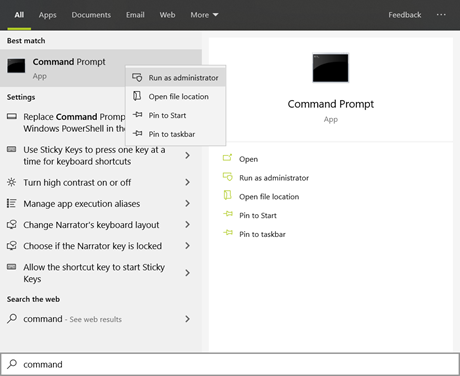


Figure . Screenshot of running “Command Prompt” with “Run as administrator”.

## ReEDS Repository Configuration

The ReEDS source code is hosted on GitHub: https://github.com/NREL/ReEDS\_OpenAccess

1. Request access to the ReEDS GitHub repository at <https://nrel.gov/analysis/reeds/request-access.html>.
2. Clone the ReEDS-2.0 repository on your desktop and use the repository with GitHub Desktop. Alternatively, download a ZIP from GitHub (Figure 9).

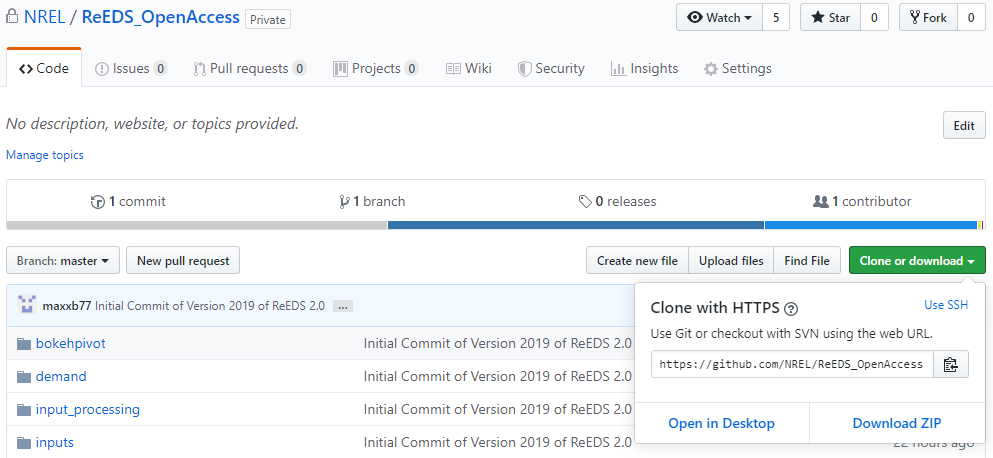


Figure 9. Screenshot of GitHub links to clone the ReEDS repository or download ZIP of the ReEDS files.

## Python Configuration

Install Anaconda: <https://www.anaconda.com/distribution/#download-section>. NREL recommends Python 3.7, but has also used Python 3.6.5 and 3.7.1 successfully.

**IMPORTANT**: Be sure to download the Windows version of the installer.

Add Python to the “path” environment variable

1. In the Windows start menu, search for “environment variables’ and click “Edit the system environment variables” (Figure 4). This will open the “System Properties” window (Figure 5).
2. Click the “Environment Variables” button on the bottom right of the window (Figure 5). This will open the “Environment Variables” window (Figure 6).
3. Highlight the Path variable and click “Edit” (Figure 6). This will open the “Edit environment variable” window (Figure 7).
4. Click “New” (Figure 7) and add the directory locations for \Anaconda\ and \Anaconda\Scripts to the environment path.

**IMPORTANT**: Test the Python installation from the command line by typing “python” (no quotes) in the terminal window. The Python program should initiate (Figure 10).

Install the gdxpds package from the command line by typing “pip install gdxpds” (no quotes) in the terminal window (Figure 8). The gdxpds package is required for reading GAMS Data Exchange files (.gdx) into Python.

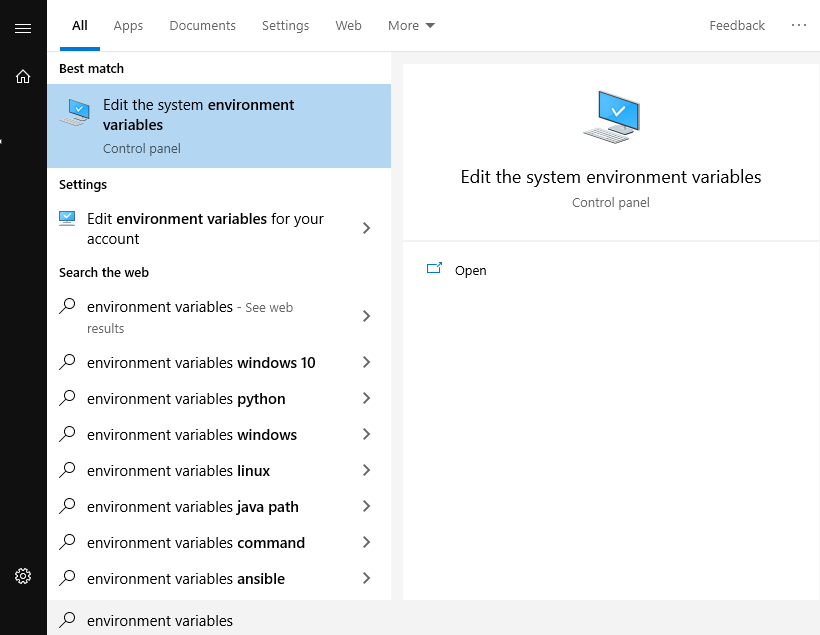


Figure 4. Screenshot of a search for “environment variables” in the Windows start menu.

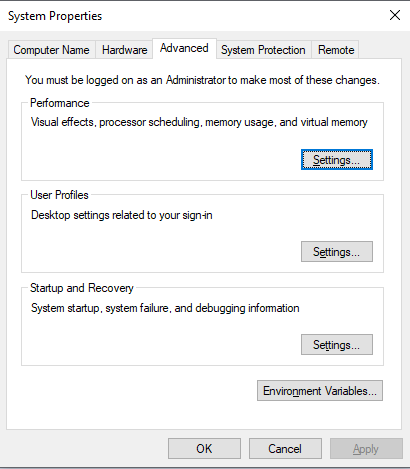


Figure 5. Screenshot of the “System Properties” window.

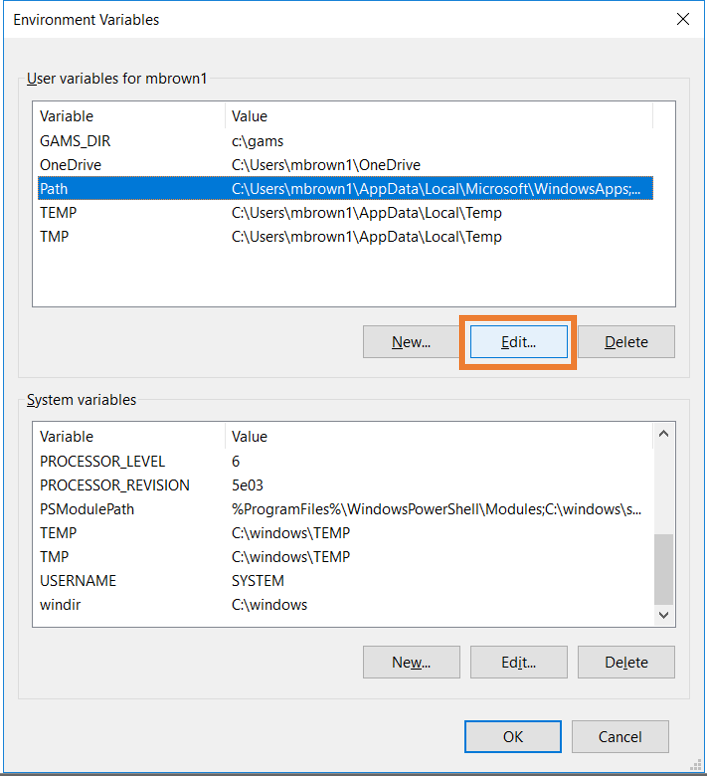


Figure 6. Edit the Path environment variable.

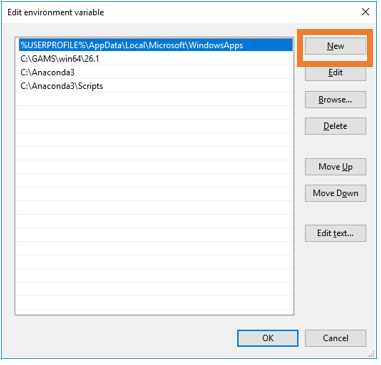


Figure 7. Append the Path environment.

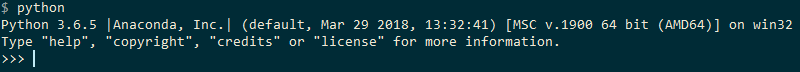


Figure 8. Screenshot of a test of Python in the terminal window.

## GAMS Configuration

Install GAMS: <https://www.gams.com/download-old/>. NREL uses GAMS version 24.7 but has not tested the compatibility of ReEDS with newer versions of GAMS. A valid GAMS license must be installed. Please refer to the “[Required Software](#_Required_Software)” section above for more information.

Add GAMS to the “path” environment variable. Follow the same instructions as for adding Python to the path in the “[Python Configuration](#_Python_Configuration)” section above. Append the environment path with the directory location for the *gams.exe* application (e.g., C:\GAMS\win64\24.7).

**IMPORTANT**: Test the GAMS installation from the command line by typing “gams” (no quotes) in the terminal window. The GAMS program should initiate (Figure 10).

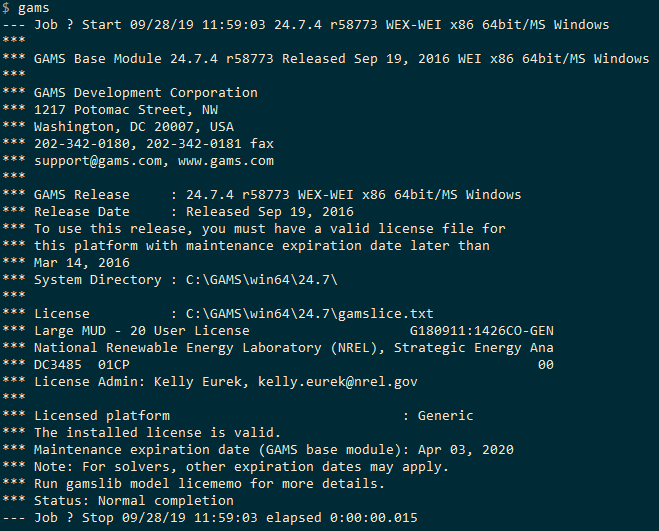


Figure . Screenshot of a test of GAMS from the terminal window.

## R Configuration

Install R 3.4.4: <https://cran.r-project.org/bin/windows/base/old/3.4.4/>. NREL has observed compatibility issues with other versions of R. NREL has not tested R versions more recent than 3.4.4. Optionally, install RStudio: <https://www.rstudio.com/products/rstudio/download/#download>.

Add R to the “path” environment variable. Follow the same instructions as for adding Python to the path in the “[Python Configuration](#_Python_Configuration)” section above. Append the environment path with the directory location for the *R.exe* and *Rscript.exe* applications (e.g., C:\Program Files\R\R-3.4.4\bin\).

**IMPORTANT**: Test the R installation from the command line by typing “r” (no quotes) in the terminal window. The R program should initiate (Figure 11).

Install R packages necessary for ReEDS from the command line. Navigate to the ReEDS directory in the terminal window. Type “rscript input\_processing\R\packagesetup.R” and press “Enter\Return”. The Rscript.exe program will install a suite of R packages (Figure 12).

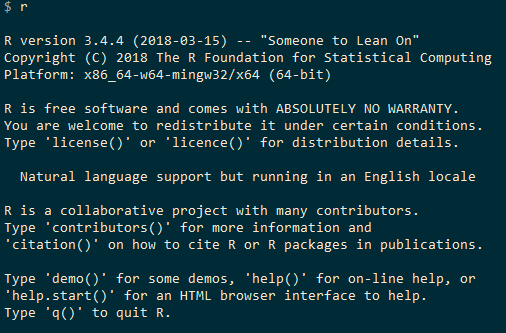


Figure . Screenshot of a test of R from the terminal window.

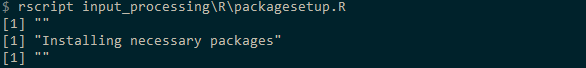


Figure . Screenshot of installing R packages with packagesetup.R.

===========================================================================

# Executing the Model

The execution of the ReEDS model can be accomplished by using a command-line interpreter application and launching a command line interface.

A ReEDS case (also referred to as a “run”, “scenario” or “instance”) is executed through a python-based case batching program called “runbatch.py”. The user can execute a single case or a batch of cases using this program.

**Step 1**: Specify the ReEDS case name(s) and configuration(s) in the case configuration file. (Figure 13). The default case configuration file name is called “cases.csv”, but the user may create custom case configuration files by using a suffix in the file name (e.g., “cases\_test.csv”). The file “case\_test.csv” can be used to execute a “test” version of the model for the ERCOT system.

Within “cases.csv”, The data in Column A are the model “switches” (also referred to as “options”). The data in Column B are brief descriptions of the switches. The data in Column C are the default values of the switches. The case configuration (or set of switches that define a case) begin with Column D. Each case configuration is represented by a single column. The case name is specified in Row 1. The value for each switch is specified beginning in Row 2. If a switch value is left blank, default value from Column C is used. A complete list of switches is provided in the [Appendix](#_ReEDS_Model_Switches) of this document.

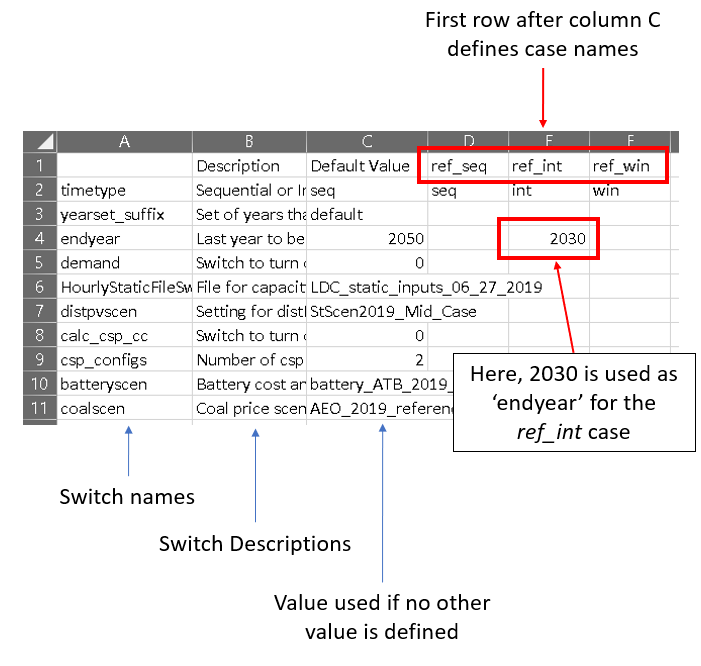


Figure 13. Screenshot of cases.csv.

**Step 2**: Initiate the case batching program from the command line

1. Initiate a command-line interpreter application and launch a command line interface.
2. Navigate to the ReEDS model directory in the command line.
3. Type “python runbatch.py” in the command line and press “Enter/Return” to initiate the ReEDS case batching program (Figure 14).
4. Provide responses to the suite of prompts in the command line (Figure 15). Please refer to the “[Prompts for user input during runbatch.py](#_Prompts_for_user)” section below for more information about the prompts.
5. Once all responses have been received, the batching program will execute the case(s) specified in the case configuration file (e.g., “cases.csv”). A separate terminal window will be launched for each case (Figure 16).

**Step 3**: Wait for each case to finish, check for successful completion, and view outputs. Once a case has finish (either from successful completion or from an error), the case-specific terminal window will close and a message in the main terminal window (i.e., where “runbatch.py” was initiated) will appear stating that the case has completed (Figure 17).

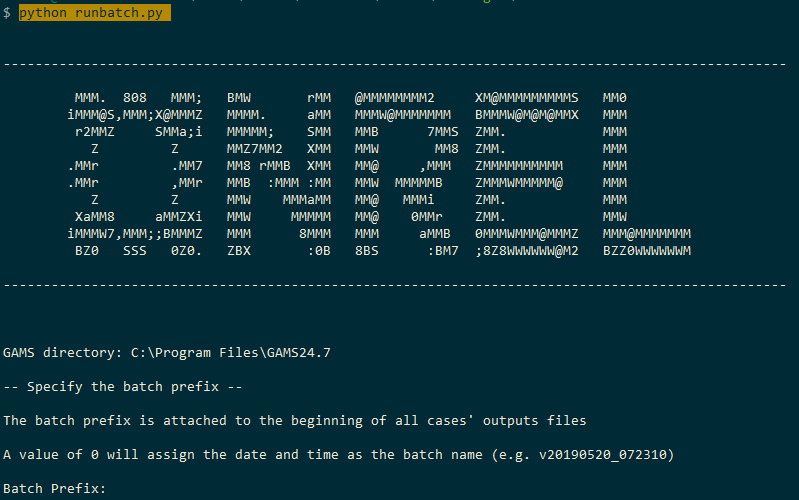


Figure 14. Screenshot of initiating “runbatch.py” from the command line

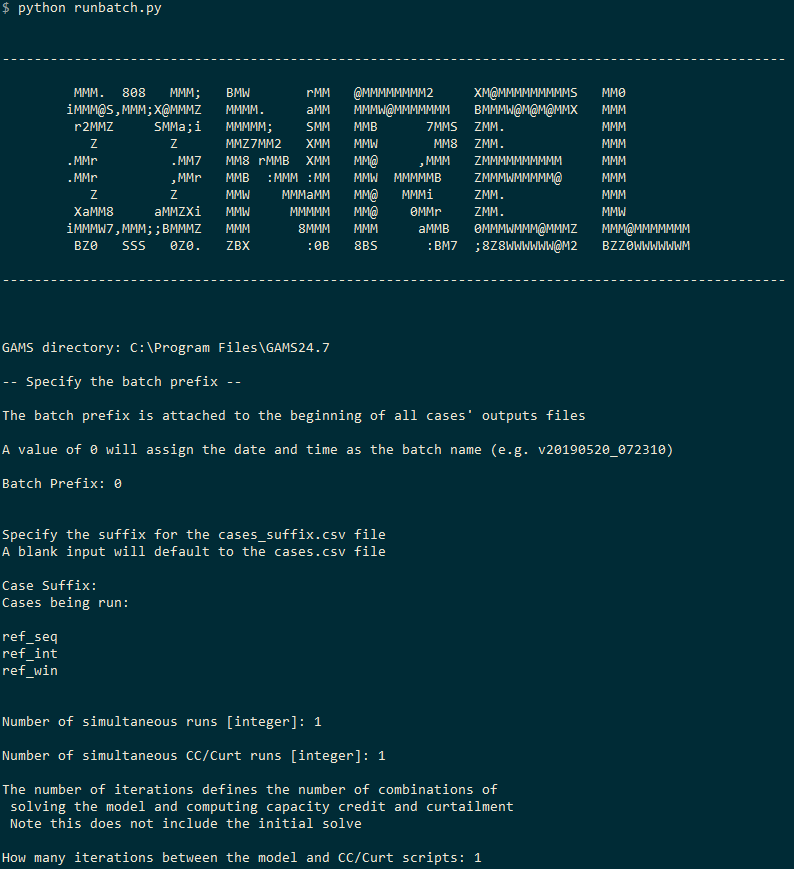


Figure 15. Screenshot of prompts for user input during “runbatch.py”.

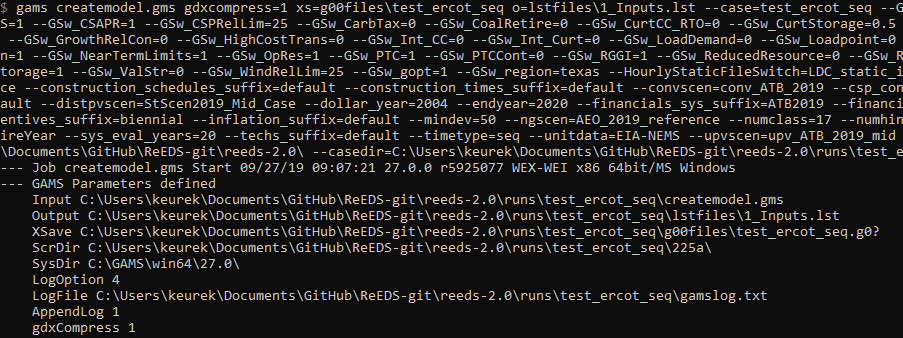


Figure 16. Screenshot of a separate terminal window being launched for a case.

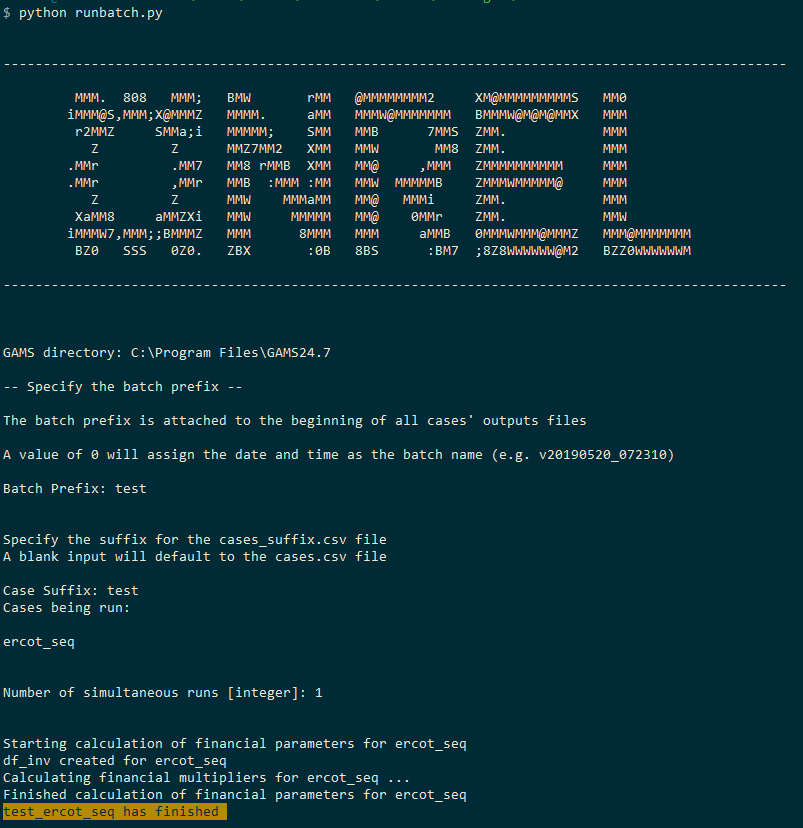


Figure 17. Screenshot of a message in the main terminal window stating when a case has finished.

## Prompts for user input during “runbatch.py”

When a user initiates a batch of ReEDS case through “runbatch.py”, a suite of prompts will appear in the terminal window. Additional details about these prompts are provided below.

**Batch Prefix[string]** – Defines the prefix for files and directories that will be created for the batch of cases to be executed (as listed in a case configuration file, e.g., “cases.csv”).

* All files and directories related to a case will be named “*{batch prefix}\_{case}”*. For example, if *batch prefix*=“test” and *case*=“ref\_seq”, then all files and directories related to this case will be named *test\_ref\_seq.* All files and directories for ReEDS cases are stored in a directory called “\runs\”
* **WARNING! A batch prefix cannot start with a number given incompatibility with GAMS.** The GAMS model declaration statement is as follows:

model {batch prefix}\_{case} /all/ ;

Therefore, “batch prefix” CANNOT begin with a numeric and SHOULD begin with an alpha character (e.g., a, A, b, B, …).

* Entering a value of “0” (zero, no quotes) will assign the current date and time for the batch prefix in the form of *v{YYYYMMDD}\_{HHMM}*. Note the preceding letter vee ‘v’ is necessary to ensure the batch prefix begins with an alpha character. For example, if *batch prefix*=“0” and *case*=“ref\_seq” on September 30, 2019 at 3:00 PM (1500 hours military time), then all files and directories related to this case will be named *v20190930\_1500\_ref\_seq*
* **WARNING! Avoid re-using a (batch prefix, case) pair**. If a directory “\runs\{batch prefix}\_{case}” already exists, a warning will be issued in the case-specific terminal window, but “runbatch.py” will overwrite data in the existing case directory (Figure 18). In some instances, the case execution will pause, and a message will appear in the case-specific terminal window “mv: replace […] overriding mode 0666?” (Figure 19). Pressing “Enter/Return” will continue the execution. NREL plans to address this overwriting issue in the future by requiring user approval to overwrite and existing case directory.

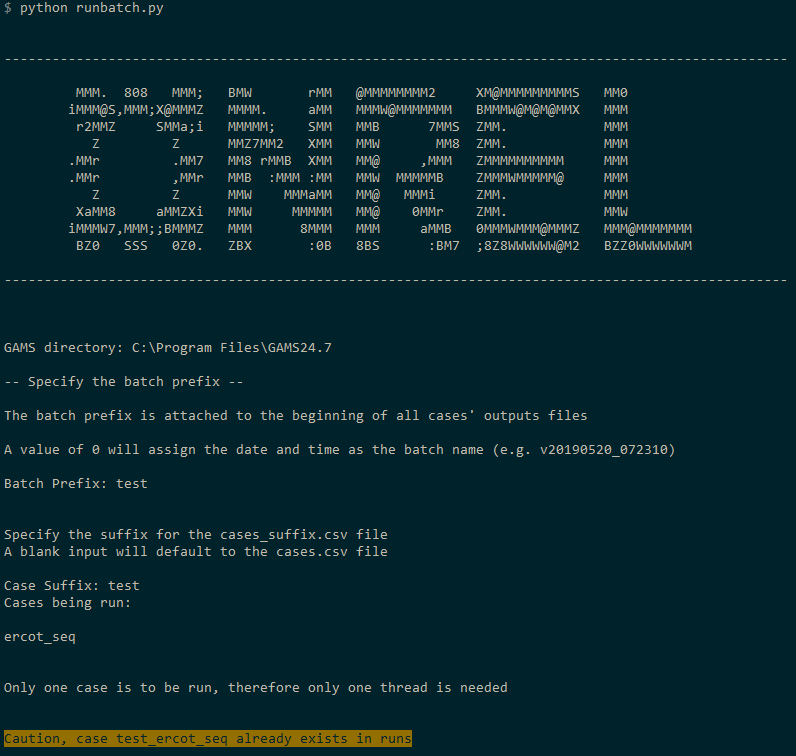


Figure 18. Screenshot of warning message that appears in the main terminal window when reusing a (batch prefix, case) pair.

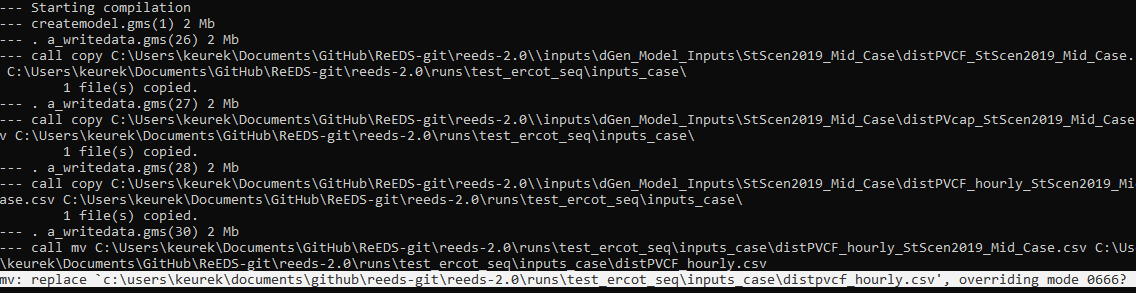


Figure 19. Screenshot of warning message that occurs in the case-specific terminal window when reusing a (batch prefix, case) pair.

**Case Suffix [string]**– Indicates which case configuration file—in the form “cases\_{case suffix}.csv” —is ingested into “runbatch.py” for processing.

* Entering an empty value (i.e., pressing “Enter/Return”) will cause the default case configuration file “cases.csv” to be used.
* **SUGGESTION**: Users may want to create a custom case configuration files (“cases\_{…}.csv”) when executing scenarios that vary from the default case configuration file (“case.csv”).

**Number of Simultaneous Runs[integer]** – Indicates how many cases should be run simultaneously in parallel.

* “runbatch.py” uses a queue to execute multiple cases.
* If there are four (4) cases and the *Number of Simultaneous Runs*=1, then “runbatch.py” will executed the cases one at a time.
* If there are four (4) cases and the *Number of Simultaneous Runs*=2, then “runbatch.py” will start two (2) cases simultaneously. Then as each case finishes a new one will start until all cases have been run.
* **WARNING**! **Be mindful about the amount of CPU and RAM usage needed for each case.**
  + Table 4 in the “[What computer hardware is necessary to run ReEDS?](#_What_computer_hardware)” section below provides some initial data points for CPU and RAM usage.
  + An Intertemporal solve will take significant resources unless simplified (through the case configuration).
  + A Sequential solve has default value of four (4) threads in “cplex.opt”. The number of threads can be reduced requiring less CPU resource usage.

**Number of simultaneous CC/Curt runs [integer]** – Indicates how many threads are to be used for the capacity credit (CC) and curtailment (curt) batching program (“reflowbatch.py”).

* This question is only asked when running intertemporal cases (i.e., timetype=“int” in the case configuration file).
* With the intertemporal case, the linear program is formulated and solved for all years at once. Then the capacity credit and curtailment calculations are executed in parallel based on the *Number of simultaneous CC/Curt runs* specified by the user.

**How many iterations between the model and CC/Curt scripts [integer]***–* For an intertemporal case, the “runbatch.py” will execute an LP solve and then call the cc/curt scripts. The value assigned here determines how many of these iterations between the LP and the cc/curt scripts occur.

* This question is only asked when running intertemporal cases (i.e., timetype=“int” in the case configuration file).
* **SUGGESTION**: When executing an intertemporal case, it is good practice to set “cc\_curt\_load = 1” in “cases.csv” to enable pre-computed starting values for capacity credit and curtailment.
* Currently, there is no convergence criteria enforced. Typically, 5-6 iterations are enough for convergence, i.e., the capacity credit and curtailment values have “small” deviations since the prior iteration.

=====================================================================================

# Runbatch.py Execution Protocol

Below are the key steps that occur when a user initiates “runbatch.py”. Knowing these steps may help the user understand the source code of “runbatch.py” if modifications need to be made to the batching program.

1. Request input from user though the command prompt.
2. Ingest case names and case switches from the user-specified case configuration file (e.g., “cases.csv”)
3. Create a new directory for each case (\runs\{batch\_prefix}\_{case}).
4. Create a batch file of execution statements for each case: (\runs\{batch\_prefix}\_{case}\call\_{batch\_prefix}\_{case}.bat)
5. All files from “filesforbatch.csv” are copied to “\runs\{batch\_prefix}\_{case}\”
   1. Any files from the root of the repository will be copied to “\runs\{batch\_prefix}\_{case}\”
   2. Any files from “\inputs\” will be copied to “\runs\{batch\_prefix}\_{case}\inputs\_case\”
6. Execute the batch file for each case in a new, case-specific window. This window will close when the case is finished. GAMS will produced files unique to each case that can help the user with error debugging:
   1. **GAMS Log File**

Path: “\runs\{batch\_prefix}\_{case}\gamslog.txt”

Purpose: contains the log outputs for all execution statements from the case batch file

**SUGGESTION**: This is a good place to check in which execution step errors may have occurred.

* 1. **GAMS listing files (\*.lst)**

Path: “\runs\{batch\_prefix}\_{case}\lstfiles\”

Purpose: contains the listing files for GAMS executions

**SUGGESTION**: This is a good place to check in which line of the source code errors may have occurred

* 1. **GAMS workfile (\*.g00)**

Path: “\runs\{batch\_prefix}\_{case}\g00files\”

Purpose: stores a snapshot all the model information available to GAMS at that point in the case execution.

For more information about GAMS work files: <https://www.gams.com/latest/docs/UG_SaveRestart.html>

**SUGGESTION**: A failed case can be restarted from this snapshot point. The user can rerun the batch file for the case (\runs\{batch\_prefix}\_{case}\call\_{batch\_prefix}\_{case}.bat) after commenting out a execution statements that completed successfully.

1. Build outputs and standard visualization reports for each case that completes successfully.
   1. Following a successful run, a suite of .csv output files are created in: “\runs\{batch\_prefix}\_{case}\outputs\”
   2. A standard .html visualization report is stored in: “\runs\{batch prefix}\_{case}\outputs\reeds-report\” contains html reports

## Case-specific Batch File Execution Protocol

**Execute CreateModel.gms:**

1. Execute a\_writedata.gms – execute several R and Python scripts and write files to “\runs\{batch prefix}\_{case}\inputs\_case\” based on selected switch settings
2. Execute B\_Inputs.gms – ingest data from the files created in a\_writedata.gms and format the data to be useful for the GAMS execution
3. Execute C\_SupplyModel.gms – declare the variables and constraints for the linear program
4. Execute C\_SupplyObjective.gms – declare the objective function for the linear program, broken into two parts (investment and operations)
5. Execute D\_SolvePrep.gms – initiate the LP solve and compute parameters based on switch values in the case configuration file (e.g., “cases.csv”)
6. Create a GAMS work file at the completion of CreateModel.gms
   1. Path: “\runs\{batch\_prefix}\_{case}\g00files\{batch\_prefix}\_{case}.g00
   2. For more information about GAMS work files: <https://www.gams.com/latest/docs/UG_SaveRestart.html>

**Execute the sequential solve structure** (Figure 20**)**

For every year in the model horizon[[7]](#footnote-7), execute d\_solveoneyear.gms:

1. Execute solve
2. Execute d\_callreflow.gms – Compute capacity credit and curtailment:
   1. Execute ReEDS\_capacity\_credit.gms - compute the parameters necessary for ReEDS\_capacity\_credit.py (which computes the capacity credit)
   2. Execute REflow\_RTO\_2\_params.gms – translate the parameters from ReEDS 2.0 into their equivalents from ReEDS Heritage
   3. Execute REflow\_RTO\_3.gms – compute curtailment using the syntax and structure from ReEDS Heritage
   4. Execute d4\_Translate\_Variability.R – Translate the values from REflow\_RTO\_3.gms (ReEDS Heritage) back to the syntax and structure used in ReEDS 2.0

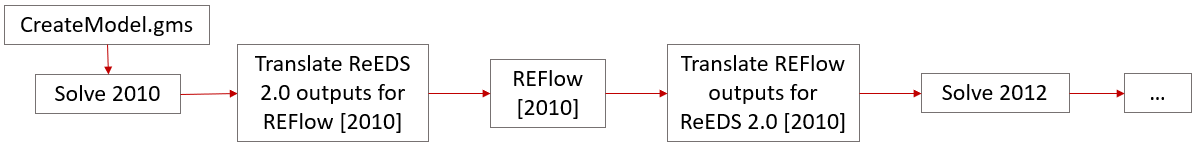


Figure 20. Depiction of execution sequence for the “sequential” solve.

**Execute the intertemporal solve structure** (Figure 21**)**

For each iteration (specified via the runbatch.py prompt),

1. Execute d\_solveallyears.gms
2. Execute reflowbatch.py
   1. Execute d\_callreflow.gms – Compute capacity credit and curtailment; same process as for sequential, but for all years
   2. Once all cc/curt calculations are done, files are merged via the GAMS gdxmerge utility
3. Execute d5\_mergevariability.R – The resulting file from the gdxmerge execution is restructured for use in GAMS during the next iteration of the intertemporal solve.

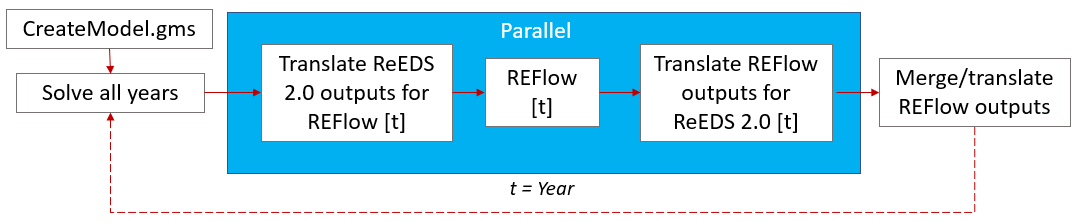


Figure 21. Depiction of execution sequence for the “intertemporal” solve.

===========================================================================

# Documentation

The ReEDS Version 2018 Documentation is available at no cost from the National Renewable Energy Laboratory: <https://www.nrel.gov/docs/fy19osti/72023.pdf>

The source code in this repository is the ReEDS Version 2019 model. Documentation for Version 2019 is forthcoming, but Table 1 summarizes difference between Versions 2018 and 2019.

Table 1. differences between ReEDS Versions 2018 and 2019.

|  |  |  |
| --- | --- | --- |
| Inputs and Treatments | 2018 Version (June 2018) | 2019 Version (July 2019) |
| Base ReEDS Model | Heritage ReEDS version | ReEDS 2.0 |
| Fuel prices | AEO 2018 | AEO 2019 |
| Demand growth | AEO 2018 | AEO 2019 |
| Generator technology cost, performance, and financing | ATB 2018 | ATB 2019 |
| Tax credit penalty | Estimated using a change in equity fraction that was proportional to the tax credit (Mai et al. 2015) | Set at 1/3 of the value of the tax credit (Bolinger 2014) |
| Wind supply curves and profiles | Based on a 2016 vintage wind turbine | Based on a 2030 vintage wind turbine |
| Existing fleet, retirements, and prescribed builds | ABB Velocity Suite from May 2018 | NEMS plant database from AEO 2019 |
| Storage capacity value | Varies based on storage and PV penetration (Frew 2018) | Varies based on load shape, wind, PV, and storage penetration |
| Storage curtailment recovery | If storage charges during a timeslice with curtailment, it reduces curtailment by one MWh for every MWh it charges | If storage charges during a timeslice with curtailment, it reduces curtailment by half a MWh for every MWh it charges |
| Transmission distances | Calculated using the straight-line distance between the geographic centroids of balancing areas (BA) | BA centroids were moved to the largest population center within the BA. Distances between these new centroids were calculated by tracing actual transmission pathways that connected the centroids. |
| AC-DC-AC interties | Existing interties represented and allowed to be expanded | Existing interties represented, but expansion of interties is not allowed |
| Clean energy policies | Not included | Included for California, Washington, New Mexico, and the Xcel portion of Colorado |
| Renewable portfolio standards and carveouts | Updated as of May 2018 | Updated as of July 31, 2019 |
| State storage mandates | Updated as of May 2018 | Updated as of July 31, 2019 |
| Canadian imports | Set exogenously based on Canada’s Energy Future 2016 (NEB 2016) | Set exogenously based on Canada’s Energy Future 2018 (NEB 2018) |
| Thermal unit representation | Coal units grouped into 4 bins per BA, all other units grouped into a single bin per BA, with representative costs and heat rates per bin | All units represented with costs and heat rates taken from the AEO 2019 NEMS database |
| Planning reserve margin | Planning reserve margin ramped down from current levels to NERC reference levels by 2025 | Planning reserve margin set at NERC reference levels for all years, except ERCOT in 2018 and 2019 is set to actual values because the actuals were lower than the NERC reference levels |
| NOx ozone season limits | Not represented | Included |

References

Bolinger, Mark. 2014. “An Analysis of the Costs, Benefits, and Implications of Different Approaches to Capturing the Value of Renewable Energy Tax Incentives.” LBNL-6610E. Berkeley, CA: Lawrence Berkeley National Laboratory. <https://emp.lbl.gov/publications/analysis-costs-benefits-and>.

Frew, Bethany A. 2018. “Impact of Dynamic Storage Capacity Valuation in Capacity Expansion Models.” NREL/PR-6A20-71858. Golden, CO: National Renewable Energy Laboratory. https://www.nrel.gov/docs/fy18osti/71858.pdf.

Mai, Trieu, Wesley Cole, Venkat Krishnana, and Mark Bolinger. 2015. “Impact of Federal Tax Policy on Utility-Scale Solar Deployment Given Financing Interactions.” NREL/PR-6A20-65014. Golden, CO: National Renewable Energy Laboratory.

NEB. 2016. “Canada’s Energy Futures 2016: Energy Supply and Demand Projections through 2040.” NE2–12/2015E–PDF. National Energy Board. http://www.neb-one.gc.ca/nrg/ntgrtd/ftr/2016/2016nrgftr-eng.pdf.

NEB 2018. “Canada’s Energy Futures 2018: Energy Supply and Demand Projections through 2040.” NE2–12/2015E–PDF. National Energy Board. https://www.neb-one.gc.ca/nrg/ntgrtd/ftr/2018/index-eng.html.

===========================================================================

# Model Architecture

## Modules

The ReEDS model is comprised of several modules with one and two-way data exchange between the modules. Figure 22 depicts these modules, including linkages between the modules and directions of data exchange. The supply module is the core module for ReEDS. Within a ReEDS execution, the key data exchanges occur between (1) the Supply Module and the Variable Resource Renewable (VRR) Modules for estimating Capacity Credit and Curtailment; and (2) the Supply Module and the Demand Module. These module interactions are dictated by the model execution approach, i.e., *sequential* solves, *sliding window* solves, or *intertemporal* solves. Figure 23 illustrates the *sequential* approach; Figure 24 the sliding window and intertemporal approaches.

Figure 22. Depiction of ReEDS modules; arrows indicate directions of data exchange.

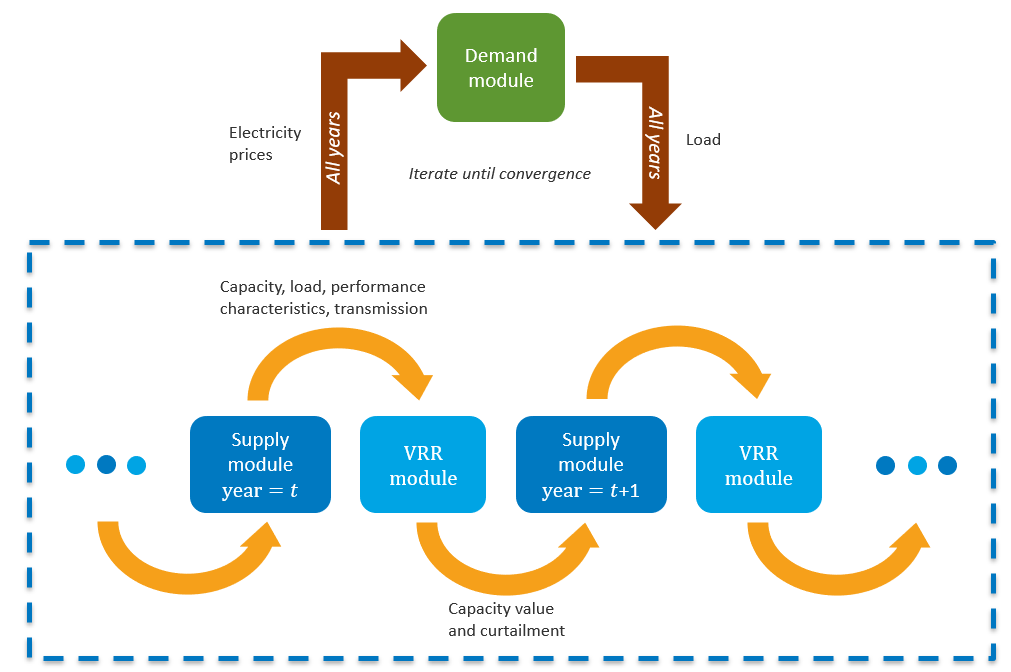


Figure 23. Schematic illustrating the model structure with a sequential solve.

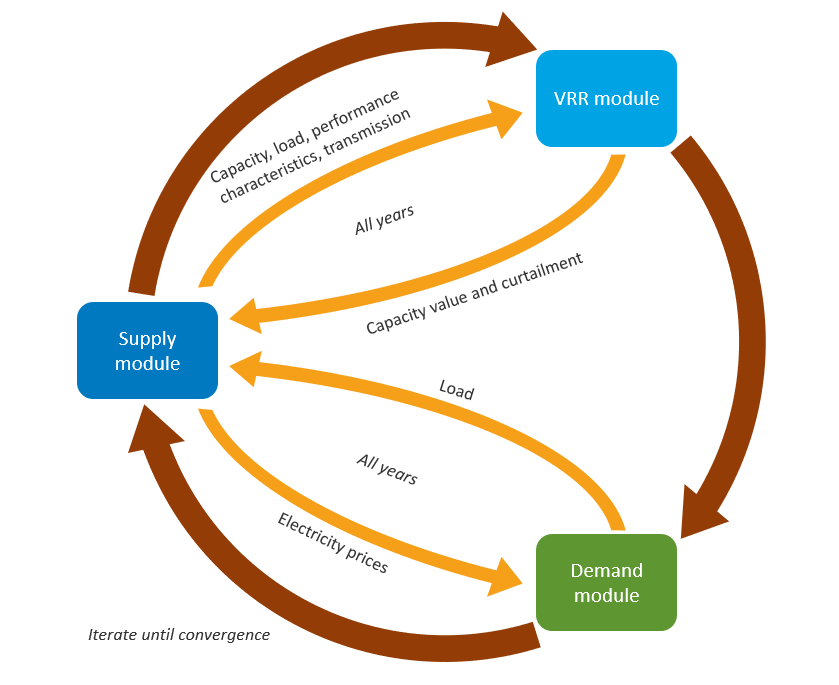


Figure 24. Schematic illustrating the model structure with sliding window or intertemporal solves.

## Tracking Capital Stock

Because ReEDS is a long-term capacity planning model, electricity generation capacity (capital stock) must be tracked over time, including, initial capacity, new investments, refurbishment investments, lifetime retirements, and endogenous retirements. Figure 25 depicts time resolution terminology and capital stock terminology. “Historical” years are 2010-2018, inclusive. “Future” years are 2019 and beyond. “Pre-modeled” years are years prior to 2010 and are not represented in the model decision making. “Modeled” years are years beginning in 2010, the first year of the model to the end of the model horizon. Users can specify the frequency of modeled years as depicted in Figure 26.

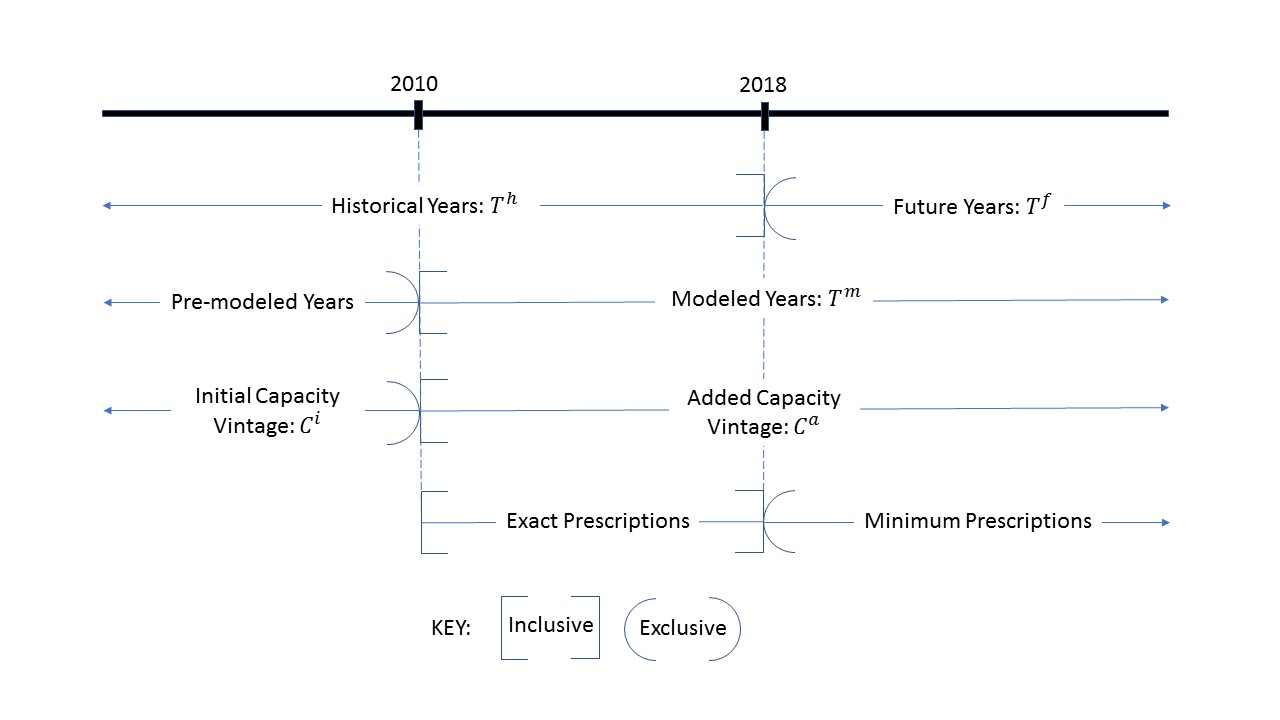


Figure 25. Depiction of time resolution terminology and capital stock terminology

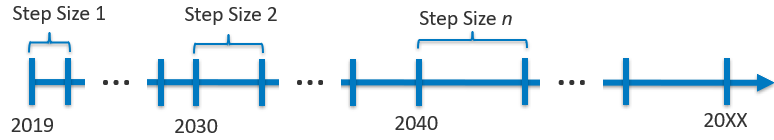


Figure 26. Depiction of user-specified model years.

Capital stock added during the model execution is tracked by a vintage classification based on the time frame when the capacity is installed (e.g., capacity installed between 2030 and 2035 could be defined as a vintage class). The user can specify the vintage class resolution by technology type to control the model size (see “\inputs\userinput\ict.csv”). All capacity of a technology type (e.g., nuclear) within the same vintage class has the same operating characteristics. Regardless of the vintage class, new investments incur capital cost associated with the year that the investment occurs.

Distinction is made between vintage classes for capacity built during the *pre-modeled* years---i.e., the *initial* vintage classes---and vintage classes for capacity built during the *modeled* years, i.e., the *added* vintage classes, (Figure 25). The *initial* vintage classes are categorized based on plant performance, specifically, heat rate (Figure 27). Whereas, *added* vintage classes are categorized based on when the capacity is installed.

*Initial* capacity is track based on the capacity remaining in each model year after planned retirements have been removed. Planned capacity additions made during *historical modeled* years, i.e., 2010-2018, are prescribed *exactly*, thus the model must build *exactly* the prescribed amount. Planned capacity built during *future modeled years*, i.e., 2019-2050, are prescribed as a *lower bound* for new investments, thus the model must build *at least* the prescribed amount (Figure 25).

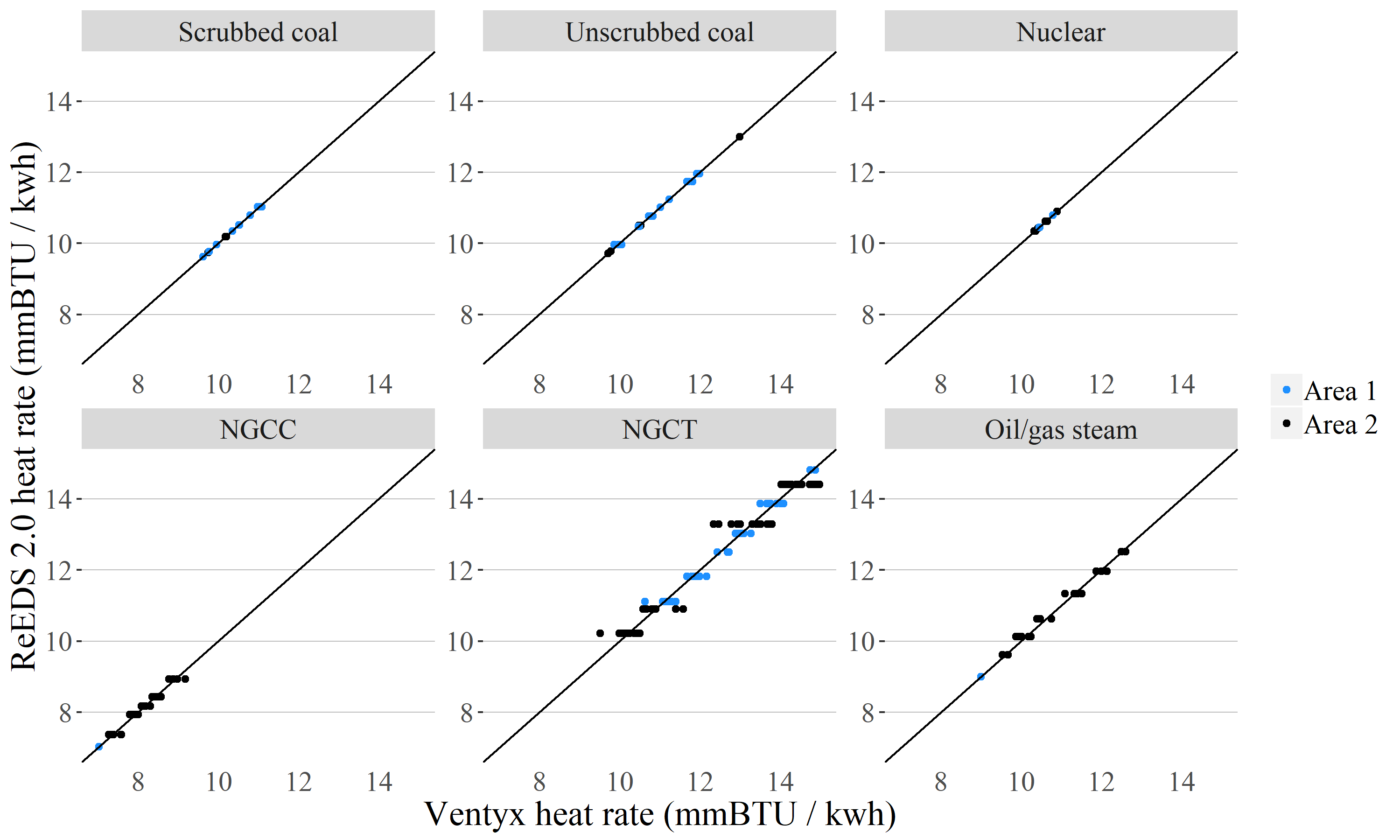


Figure 27. Example of categorizing model plants for the existing fleet based on heat rate.

===========================================================================

# Frequently Asked Questions

## How much are the GAMS licensing fees?

The GAMS licensing price list is available on the GAMS website (<https://www.gams.com/products/buy-gams/>). Prices listed are “for an unrestricted, perpetual, single named user on a specific platform.” The prices for additional users are a function of the single user price. The tables below are examples of the pricing for a multi-user licenses for the Base Module and GAMS\CPLEX .

Table 2. Example of initial GAMS licensing fees (USD) based on the number of users for a **Standard** License on a single platform (e.g., Windows).

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Users | GAMS Base Module | GAMS\CPLEX | Total |
| 1 | 3,200 | 9,600 | 12,800 |
| 5 | 6,400 | 19,200 | 25,600 |
| 10 | 9,600 | 28,800 | 38,400 |
| 20 | 12,800 | 38,400 | 51,200 |
| 30 | 16,000 | 48,000 | 64,000 |

Table 3. Example of initial GAMS licensing fee example (USD) based on the number of users for an **Academic** License on a single platform (e.g., Windows).

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Users | GAMS Base Module | GAMS\CPLEX | Total |
| 1 | 640 | 1,280 | 1,920 |
| 5 | 1,280 | 2,560 | 3,840 |
| 10 | 1,920 | 3,840 | 5,760 |
| 20 | 2,560 | 5,120 | 7,680 |
| 30 | 3,200 | 6,400 | 9,600 |

## Is there a trial version of the GAMS license so that I can test ReEDS?

You *may* be able to request a temporary evaluation licenses of the GAMS base module and a GAMS\Solver license. Please contact GAMS for more information.

## What computer hardware is necessary to run ReEDS?

NREL uses Windows servers to execute the ReEDS model. These servers have Intel(R) Xeon(R) CPUs at 2-2.4GHz and 10-14 cores with 160-320GB of RAM. These servers can execute multiple ReEDS cases in parallel.

The table below summarizes RAM usage and total clock time necessary to execute the default ReEDS configuration for the three different solve structures (sequential, window, and intertemporal) using one of these servers.

Table 4. Ballpark RAM usage and clock time for select model configurations executed on NREL servers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solve Structure | Solve Steps | Approximate LP size per instance (after presolve) | Threads | RAM  (GB per instance) | Total Clock Time (hours) |
| Sequential | 2010-2030: 2 yr  2031-2050: 5 yr | Rows: 461k  Columns: 384k  Non-Zeros: 2,042k | 4 | 4-7GB | 3-5 |
| Window | 2010-2030: 2 yr  2031-2050: 5 yr | Rows: 1,143k  Columns:869k  Non-Zeros: 5,087ka | 8 | 14-18GB | 18-22 |
| Intertemporal | 2010-2030: 2 yr  2031-2050: 5 yr | Rows:12,789k  Columns: 9,890k  Non-Zeros: 61,076k | 12 | 57GB+ | 48-72b |

a Default solve window configuration: 11 windows and 5 iterations per window (see “\inputs\userinput\windows.csv”). The problem size is taken from last iteration. The problem size varies by about 5% across iterations.

b 8-12 hours per iteration, 5-7 iterations total

===========================================================================

# Contact Us:

If you have comments and/or questions, please contacts the ReEDS team:

[ReEDS.Inquiries@nrel.gov](mailto:ReEDS.Inquiries@nrel.gov)

===========================================================================

# Appendix:

## ReEDS Model Switches (specified by user in “cases.csv”)

### Modeling horizon switches

**timetype** [string] - defines how the model portrays foresight

* **seq** (sequential) solves one solve year at a time assuming the modeled year’s operations last for 20 years
* **int** (intertemporal): full foresight of all modeled years
* **win** (window): foresight for a selected period of time, defined in ..\inputs\userinput\windows.csv

default value: **seq**

**yearset\_suffix** [string]– file pointer suffix used in the file path to specify the model years

path: \inputs\userinput\modeledyears\_{suffix}.csv

default value: **default**

**endyear** [integer] - Last year to be modeled

default value: **2050**

### Demand switches

**demand** [binary] **–** switch to turn on demand module

* 0 – Exclude demand module (OFF)
* 1 – Include demand module (ON)

default value: 0 (OFF)

**demandscen** [string] – file pointer for electricity demand profile

path: \inputs\loaddata\demand\_{demandscen}.csv

default value: AEO\_2019\_reference

**GSw\_LoadDemand** [binary] – Set inelastic demand when running the demand side

Default value: 0

**GSw\_EV** [binary] – Turn electric vehicle demand on/off

Default value: 0

### Renewable energy capacity credit and curtailment switches

**HourlyStaticFileSwitch** [string] – file pointer for hourly data used for capacity credit calculations

path: \inputs\variability\{HourlyStaticFileSwitch}.csv

default value: LDC\_static\_inputs\_06\_27\_2019

**calc\_csp\_cc** [binary] –Turn on/off CSP capacity credit calculations

* 0 – OFF
* 1 – ON

default value: 0 (OFF)

**csp\_configs** [integer] – number of CSP configurations for hourly calculations

default value: 2

**cc\_curt\_load** [binary] – Switch to enable loading of default capacity credit and curtailment parameters in the intertemporal case (stored in inputs\cccurt\_defaults.gdx). When executing an intertemporal case, it is good practice to set “cc\_curt\_load = 1” in “cases.csv” to enable pre-computed starting values for capacity credit and curtailment.

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_AVG\_iter** [integer] – Select method for choosing which iterations are used for CC/curt calculations

Default value: 1

**GSw\_CurtCC\_RTO** [binary] – Switch to average cc curt values over RTO

Default value: 0

**GSw\_Int\_CC** [integer] – Select intertemporal capacity credit method

* 0=average undifferentiated
* 1=average differentiated
* 2=marginal differentiated

Default value: 0

**GSw\_Int\_Curt** [integer] – select intertemporal curtailment method

* 0=average undifferentiated
* 1=average differentiated
* 2=marginal undifferentiated
* 3=marginal differentiated

Default value: 0

**GSw\_CurtStorage** [fraction] – fraction of storage charging that can reduce curtailment

Default value: 0.5

### Technology cost and performance switches

**distpvscen** [string] – file pointer for rooftop PV deployment trajectories from the dGen model

path: \inputs\dGen\_Model\_Inputs\{dispvscen}.csv

default value: StScen2019\_Mid\_Case

**batteryscen** [string] – file pointer for battery cost and performance characteristics

path: \inputs\plant\_characteristics\{batteryscen}.csv

default value: battery\_ATB\_2019\_mid

**coalscen** [string] – file pointer for battery cost and performance characteristics

path: \inputs\fuelprices\coal\_{coalscen}.csv

default value: AEO\_2019\_reference

**convscen** [string] – file pointer for conventional generation cost and performance characteristics

path: \inputs\plant\_characteristics\{convscen}.csv

default value: conv\_ATB\_2019

**cspscen** [string] – file pointer for CSP cost and performance characteristics

path: \inputs\plant\_characteristics\{cspscen}.csv

default value: csp\_ATB\_2019\_mid

**geoscen** [string] – file pointer for geothermal cost and performance characteristics

path: \inputs\plant\_characteristics\{cspscen}.csv

default value: geo\_ATB\_2019\_mid

**geosupplycurve** [string]– file pointer for geothermal resource supply curve

path: \inputs\geothermal\geo\_rsc\_{geosupplycurve}.csv

default value: BAU

**hydroscen** [string] – file pointer for hydro power cost and performance data

path: \inputs\plant\_characteristics\{hydroscen}.csv

default value: hydro\_ATB\_2019\_mid

**ngscen** [string] – file pointer for natural gas prices

path: \inputs\fuelprices\ng\_{ngscen}.csv

default value: AEO\_2019\_reference

**retscen** [string] – generator retirement schedule

path to generator database: \inputs\capacitydata\ReEDS\_generator\_database\_final\_EIA-NEMS.csv

default value: NukeRefRetireYear

**upvscen** [string] – file pointer for UPV cost and performance data

path: \inputs\plant\_characteristics\{upvscen}.csv

default value: upv\_ATB\_2019\_mid

**uraniumscen** [string] – file pointer for uranium prices

path: \inputs\fuelprices\uranium\_{uraniumscen}.csv

default value: AEO\_2019\_refernce

**windscen** [string] – file pointer for UPV cost and performance data

path: \inputs\plant\_characteristics\{windscen}.csv

default value: wind\_ATB\_2019\_mid

### Region switches

The following three switches are interrelated and are used together to gather the regional information for a scenario:

Select all rows in “\inputs\regions\regions\_{region\_suffix}.csv” where the entry for column “{region\_type}” is equal to “{GSw\_region}.”

Example for the lower 48 US:

Select all rows in “\inputs\regions\regions\_default.csv” where the entry for column “country” is equal to “usa.”

Example for ERCOT:

Select all rows in “\inputs\regions\regions\_ercot.csv” where the entry for column “interconnect” is equal to “Texas.”

An example for ERCOT is shown in “cases\_test.csv”

**regions\_suffix** [string] – file pointer suffix for regional hierarchy file

path: \inputs\regions\regions\_{regions\_suffix}.csv

* default (for lower 48 US analysis)
* ercot (for ERCOT analysis)

Default value: default

**region\_type** [string] – specify what region type will be used for “GSwS\_region”. Valid inputs are either column names in regions file or 'custom' (e.g. naris), where custom regions are defined in their own file. Model will not work for all regions.

* country (for lower 48 US analysis)
* interconnect (for ERCOT analysis)

Default value: country

**GSw\_region** [string] – spatial extent of model regions

* usa (lower 48 US analysis)
* texas (for ERCOT analysis)

Default value: usa

### Financing Switches

**construction\_schedules\_suffix** [string] – file pointer suffix for construction schedules

Default value: default

**construction\_times\_suffix** [string] – file pointer suffix for construction times by technology

Default value: default

**depreciation\_schedules\_suffix** [string] – file pointer suffix depreciation schedules

Default value: default

**financials\_sys\_suffix** [string] – file pointer suffix for the system-wide system discount rate

Default value: ATB2019

**financials\_tech\_suffix** [string] – file pointer suffix for technology-specific financial assumptions

Default value: ATB2019\_mid

**incentives\_suffix** [string] – file pointer suffix for incentive definition

Default value: biennial

**inflation\_suffix** [string] – file pointer suffix for historical inflation schedule

Default value: default

**reg\_cap\_cost\_mult\_suffix** [string] – file pointer suffix for regional capital cost multipliers

Default value: default

**techs\_suffix** [string] – file pointer suffix for master list of technologies

path: \inputs\techs\techs\_{techs\_suffix}.csv

Default value: default

**dollar\_year** [integer] – DO NOT CHANGE FROM 2004 UNTIL ALL FINANCIAL INPUTS HAVE DOLLAR YEAR ADJUSTMENT – Real dollar year for model to calculate and report

Default value: 2004

**sys\_eval\_years** [integer] – Number of years that the model evaluates investments on

Default value: 20

### Model plant switches

**Numclass**  [integer] – maximum number of vintage classes for generating capacity added after 2010

Default value: 17

**Numhintage** [integer] – maximum number of vintage bins for generating capacity existing prior to 2010; specifying “unit” will give a unit-level representation

Default value: 6

**mindev** [string]- minimum heat rate deviation for binning existing (prior to 2010) generating units into a separate vintage bin

Default value: 50

**Unitdata** [string] – pointer for which unit database to use

* "EIA-NEMS" – use the unit database from the EIA’s NEMS model

Default value: EIA-NEMS

**GSw\_BinOM** [binary] – Turn on/off binned FOM and VOM for each historical vintage bin

* 0 – OFF
* 1 – ON

Default value: 1 (ON)

### Capital stock switches

**GSw\_ForcePrescription** [binary] – Turn on/off forced prescriptions - turning off will allow unlimited but not free builds in historical years

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_Refurb** [binary] – Turn on/off refurbishments

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_Retire** [binary] – Turn on/off endogenous retirement

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_RetireYear** [integer] – Year to begin retirements

Default value: 2022

**GSw\_CoalRetire** [binary] – Adjust lifetime coal retirements

* 0 – OFF
* 1 – ON

Default value: 0

### Capacity growth limit switches

**GSw\_GrowthAbsCon** [binary] – Turn on/off absolute growth constraint

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_GrowthRelCon** [binary] – Turn on/off relative growth constraint

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_CSPRelLim** [percent] – Value of the CSP relative limit (% / year)

Default value: 25

**GSw\_SolarRelLim** [percent] – Value of the solar relative limit (% / year)

Default value: 25

**GSw\_WindRelLim** [percent] – Value of the wind relative limit (% / year)

Default value: 25

**GSw\_NearTermLimits** [binary] – Turn on/off near term capacity investment decisions

* 0 – OFF
* 1 – ON

Default value: 1

### Policy switches

**GSw\_AB32** [binary] – Turn on/off AB32

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_AnnualCap** [binary] – Turn on/off CO2 cap

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_BankBorrowCap** [binary] – Turn on/off CO2 cap with banking and borrowing

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_BatteryMandate** [binary] – Turn on/off battery mandate constraint

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_CSAPR** [binary] – Turn on/off the CSAPR emissions regulation

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_CarbTax** [binary] – Turn on/off CO2 tax

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_PTC** [binary] – Switch to turn on/off wind PTC

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_PTCCont** [binary] – Extend PTC for wind

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_RGGI** [binary] – Turn on/off RGGI

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_StateRPS** [binary] – Turn on/off state RPS requirements

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_MCS** [string] – Select countries to apply the mid-century strategy

Default value: no

**GSw\_GenMandate** [binary] – Turn on/off national Gen Requirement

* 0 – OFF
* 1 – ON

Default value: 0

### Technology inclusion switches

**GSw\_Geothermal** [integer] – inclusion of geothermal

* 0 – OFF
* 1 – ON; default representation
* 2 – ON; extended representation

Default value: 1

**GSw\_Storage** [binary] – Turn on/off all storage

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_CCS** [binary] – Turn on/off all Carbon Capture and Storage (CCS) technologies

* 0 – OFF
* 1 – ON

Default value: 1

### Other model constraint switches

**GSw\_GasCurve** [integer] – Select natural gas supply curve

Default value: 0

**GSw\_HighCostTrans** [binary] – Turn on higher cost and higher losses transmission

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_MaxCFCon** [binary] – Turn on/off minimum seasonal CF constraint

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_MinCFCon** [binary] – Turn on/off min CF constraint

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_Mingen** [binary] – Turn on/off Mingen variable

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_OpRes** [binary] – Turn on/off operating reserve constraints

* 0 – OFF
* 1 – ON

Default value: 1

**GSw\_ReducedResource** [binary] – Turn on/off switch to reduce the RE resource available

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_ReserveMargin** [binary] – Turn on/off planning reserve margin

* 0 – OFF
* 1 – ON

Default value: 1

### Linear programming switches

**GSw\_Loadpoint** [binary] – Turn on/off the use a GAMS “loadpoint” for the intertemporal case

* 0 – OFF
* 1 – ON

Default value: 0

**GSw\_gopt** – select solver option file to be used

Default value: 1

**GSw\_ValStr** – Turn on/off value stream calculation; this is a decomposition of reduced cost to help understand the model decision making

* 0 – OFF
* 1 – ON

Default value: 0

1. <https://www.gams.com/latest/docs/UG_License.html> [↑](#footnote-ref-1)
2. <https://www.gams.com/products/buy-gams/> [↑](#footnote-ref-2)
3. <https://wiki.python.org/moin/BeginnersGuide/Overview> [↑](#footnote-ref-3)
4. <https://www.python.org/about/> [↑](#footnote-ref-4)
5. <https://docs.conda.io/en/latest/> [↑](#footnote-ref-5)
6. <https://www.r-project.org/about.html> [↑](#footnote-ref-6)
7. modeled years are specified in “\inputs\user\_input\modeledyears\_default.csv” and the horizon is specified in “cases.csv” [↑](#footnote-ref-7)