# Switch 2.0 Tutorial: Pre-Tutorial Setup

Matthias Fripp
Assoc. Prof. of Electrical Engineering
University of Hawaii, Manoa
mfripp@hawaii.edu

Before you come for the tutorial, please complete the steps described here to install Switch on a laptop computer and learn some background information. Section 2.2 and 2.3 are optional. You will need an Internet connection while you do these. Required steps are marked by a blue highlight in the margin like this paragraph. Explanatory text and optional steps don't have this mark.

## 1 Software setup (approx. 1 hour; requires 4-5 GB disk space)

This section describes how to install Switch on your computer so it is ready to solve power system planning problems. Switch depends on a collection of mostly open-source software:

- **Switch core modules** and **your extension modules** define a power system optimization model in relatively easy-to-understand Python code.
- **Switch** loads the modules you have selected and the corresponding data, constructs an optimization model, solves it and reports results.
- **Pyomo** is a general-purpose optimization modeling framework for Python. Switch uses this package to define the elements of your optimization model and call a solver.
- Pyomo converts the Switch model into a standardized, computer-readable form and sends it to an **external solver** (e.g., glpk, cbc, cplex or gurobi). The external solver does the intense computation required to find an optimal plan.
- **Anaconda** provides an easy, standardized way to install Python, Pyomo, glpk and other tools on any computer system.

The instructions below will show you how to setup the following:

- Anaconda Python environment
- Switch and software it depends on (Pyomo and glpk)
- Example models to use in this tutorial
- Atom text editor

All of these tools are open-source and cross-platform, so you should be able to use them on any computer.

Disk usage: Switch itself is quite small, but it depends on the Anaconda distribution (0.3-0.5 GB), and a number of Python packages (0.5-1.5 GB). We will also use the Atom text editor for this tutorial (0.8 GB), and install some tutorial data (0.3 GB). Note that the higher disk usage numbers are for Windows installations.

## 1.1 Installing Anaconda and Python

Download and install the "mini" version of Anaconda from https://docs.conda.io/en/latest/miniconda.html. I recommend selecting the 64-bit Python 3+ version (other versions will probably work too). On a Mac, the ".pkg" installer is easier to use than the "bash" one. Note that you do not need administrator privileges to install Anaconda or add packages to it if you choose the "Install for: Just Me" option during installation.

(You can also use the "full" version of Anaconda from https://www.anaconda.com/distribution/, but that requires 2–3 GB more disk space).

## 1.2 Installing Switch, Pyomo and glpk

Open Terminal.app (OS X: in Applications/Utilities folder) or an Anaconda command prompt (Windows: Start > Anaconda3 > Anaconda Command Prompt). Then type this command and press Enter or return:

```
conda install -c conda-forge switch_model
```

Follow the prompts to install Switch and related software.

You can check that everything has been installed by running each of these commands:

```
glpsol --version
python --version
pyomo --version
switch --version
```

You should see version numbers for each one, and no errors.

## 1.3 Retrieving tutorial data and Switch source code

Open Terminal.app or an Anaconda command prompt if you don't have it open already. Then type this command and press Enter or return:

```
conda install git
```

Follow the prompts to install "git". On a Mac, if you are prompted to install the command-line developer tools, click "Install".

Use the "cd" and "mkdir" commands navigate to a good location to install the tutorial data (e.g., run "cd Documents" then "mkdir tutorial" then "cd tutorial", and the data will be placed inside a "tutorial" folder inside your "Documents" folder). Then use the following command to copy the tutorial data to a subdirectory called "switch_tutorial" within the current directory:

```
git clone --depth=1 https://github.com/switch-model/switch_tutorial.git
```

> Now use the following command to copy the source code for Switch into a subdirectory called “switch” within the current directory. This will be useful for inspecting the source code later.

```
git clone --depth=1 https://github.com/switch-model/switch.git
```

Note: you now have two copies of Switch on your system—one for running and one for viewing. It is possible to use one copy for both, but it is a little tricky. If you want to do that, please see the last section of https://github.com/switch-model/switch/blob/master/INSTALL.md.

## 1.4 Installing Atom text editor and tablr extension

For this tutorial, I assume you are using the Atom text editor to view and edit code and data files. You can use a different text editor if you like, but it should be capable of doing programming-oriented tasks, like quickly adjusting the indentation of groups of lines in a text file. If you prefer, you can also open the .csv data files directly in your spreadsheet software instead of using Atom and the tablr extension.

> Download and install the Atom text editor from https://atom.io.

> **On a Mac**, you should finish the installation by copying the Atom app from your download folder to your Applications folder.

**On Windows,** Atom uses a standard installer, but it may spend a few minutes on the animated installer screen.

If you need more info on installing Atom, see https://flight-manual.atom.io/getting-started/sections/installing-atom/.

If you'd like a quick intro to the program, see https://flight-manual.atom.io/getting-started/sections/atom-basics/.

> Next, open Atom from the Start menu or Applications folder. View and/or close the introductory tabs to clean up the display a little.
>
> Now you will install the “atom-tablr” package, which allows you to easily view and edit tabular text files, such as the .csv files used by Switch. This step is optional, but it will make the .csv files much easier to read.
>
> The atom-tablr package is great, but unfortunately slightly broken and no longer maintained. However, you can install a fixed version as follows: In Atom, choose the Atom > Preferences

> menu (Mac) or File > Settings menu (Windows). Then click on the "+ Install" tab. In the "Search packages" box at the top, type "https://github.com/mfripp/atom-tablr.git" (without the quotes). Below this, you should see a box that says "git+ https://github.com/mfripp/atom-tablr.git". Click on the "Install" button in this box. It should take a minute or so to install, and then the "Install" button will change to "Settings", "Uninstall" and "Disable". (If this doesn't work, see footnote 1.)
>
> Next, I recommend clicking on the "Settings" button for atom-tablr and adjusting the following settings:
> Csv Editor (about 1/4 way down): turn on "File Header"
> Table Editor (about 1/2 way down): set Column Width to 180
> (On some systems the Settings button does not appear; if so, you can skip this step.)

By default, Atom uses a light-on-dark color scheme. If you'd like to change this, you can do that via the Themes tab in the Settings pane (Atom > Preferences > Themes on a Mac, File > Settings > Themes on Windows).

Optional (not used in this tutorial): For the future, I also recommend installing the "hydrogen" and "hydrogen-python" packages. These allow you to run Python code directly from Atom, which is great for testing and troubleshooting scripts as you write them (e.g., data conversion and analysis scripts). To install those, go back to Atom and open Atom > Preferences… or File > Preferences…. Choose the Install tab. Search for "hydrogen". Click the "Install" buttons for "hydrogen" and "hydrogen-python". To support these, you will also need to run these two commands (one time) in a Terminal window or Anaconda Command Prompt: "`conda install ipykernel`", then "`python -m ipykernel install --user`". You may also be interested in the Atom packages for running R, Stata or LaTeX code from the editor.

## 1.5 Installing a commercial solver (optional)

The conda command automatically installs the open-source glpk solver along with Switch. This and other open-source solvers (e.g., cbc) are able to solve small test cases, but are not fast enough to solve large models. So I generally use proprietary solvers (cplex or gurobi) in practice. These are expensive for professional use, but it is possible to get a trial license before you buy a long-term one. Academics can also get full licenses for free. Note that it is important to get a license (temporary or long-term) for the *full* version of the software, not the free or community version that only supports small problem sizes.

You can complete this tutorial using only glpk, but if you would like to experiment on your own with the larger models, you should install cplex or gurobi. You can obtain licenses and download these solvers from here:

[1] This sometimes fails on Windows. In that case, open a non-Anaconda command prompt (you can find it by searching for cmd.exe in the search bar), then run the command "apm install https://github.com/mfripp/atom-tablr.git". After it runs, restart Atom, go to File > Settings, then the Packages tab, then search for tablr among the installed packages, and continue from there.

Professional:
https://www.gurobi.com/products/gurobi-optimizer/
https://www.ibm.com/products/ilog-cplex-optimization-studio/pricing

Academic:
https://www.gurobi.com/academia/
https://developer.ibm.com/docloud/blog/2019/07/04/cplex-optimization-studio-for-students-and-academics/

Once you have installed these, you can test that they are available by running one of these pairs of commands from a command prompt or terminal window:

Gurobi on Windows:

```
gurobi.bat
exit()
```

Gurobi on Mac or Linux

```
gurobi.sh
exit()
```

CPLEX

```
cplex
quit
```

## 2 Introduction to Switch, Pyomo and Python (0.5 – 9 hours)

This section points you to some useful, quick introductions to Switch, Python and Pyomo. The latter two are optional, but recommended if you will be using Switch extensively or defining custom behaviors (new technologies, rules or policies).

### 2.1 Introduction to Switch (0.5 – 2 hours)

> For a quick overview of Switch, please read Section 2 of the paper on Switch 2.0 at https://doi.org/10.1016/j.softx.2019.100251.

The following are optional: You can read section 3 of the Switch paper for an overview of the case study we'll examine during this tutorial. And if you would like more detail on Switch, please see the Supplementary Material for the Switch 2.0 paper at https://ars.els-cdn.com/content/image/1-s2.0-S2352711018301547-mmc1.pdf.

## 2.2 Introduction to Python (optional, 1 – 6 hours)

I recommend reading sections 3 and 4 of the Python introduction at https://docs.python.org/3/tutorial/ . If you would like a deeper understanding, sections 5 and 6 are also worth reading.

If you want to run sample code from the Python tutorial, you can do so as follows: Open Terminal.app (Mac) or an Anaconda command prompt (Windows). Then type "python<enter>" to start the Python interpreter. Then copy code from the Python tutorial into the interpeter. Generally the code marked with ">>>" or "..." is code that you can type to the Python interpreter. But you shouldn't copy the >>> or ... symbols themselves.

## 2.3 Introduction to Optimization and Pyomo (optional, 1 – 3 hours)

I recommend going through the following sections at https://pyomo.readthedocs.io/ to get an introduction to Pyomo, the optimization software used by Switch.

- Pyomo Overview
- Pyomo Modeling Components
- Solving Pyomo Models

This will enable you to read and write Switch code, which is just Pyomo code applied to power system modeling. i.e., a Switch model is a Pyomo AbstractModel used to optimize the design of a power system.

**Notation:** In the Pyomo introduction, you will see problems with a notation like this:

min $c_1x_1 + c_2x_2$
s.t.
$a_{11}\ x_1 + a_{12}\ x_2 \geq b_1$
$a_{21}\ x_1 + a_{22}\ x_2 \geq b_2$
$x_1 \geq 0$
$x_2 \geq 0$

This is a common way to describe mathematical optimization problems. It means "find values for $x_1$ and $x_2$ that will minimize the value of $c_1x_1 + c_2x_2$, such that all the specified constraints are satisfied."

In this problem, the *x* values are called **decision variables** (these are numbers that will be chosen when the problem is run, e.g., the amount of power to produce from a project during a particular hour), the *a*, *b*, and *c* values are **parameters** (data you know when you set up the problem, e.g., the maintenance cost per MWh produced from a project), $c_1x_1 + c_2x_2$ is the **objective function** (the value to be minimized or maximized), and the other equations are the **constraints** (e.g., that power output is less than or equal to installed capacity).

Now you are all set for the tutorial. See you next week.