Internal Demo of the CALVIN model

1.0 Installation and set up

1. Install Anaconda to have a Python package on your computer.

The easiest way to get the python is to download Anaconda package from https://www.continumm.io/downloads. While downloading, you need to make sure that the version of Python 3 or later. Once you download the package, install Anaconda following the prompts that appeared on the screen.

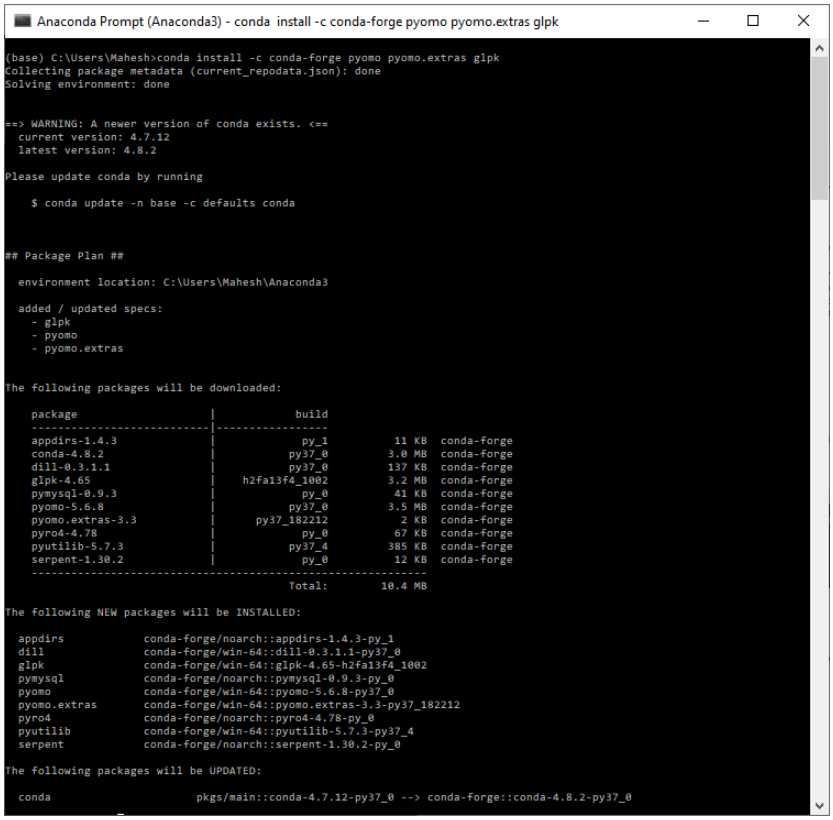
2. After installing Anaconda, you need to install Pyomo. Window users shall go to the Anaconda command prompt (Figure 1) in the following order:
    a. Click Start
    b. Navigate Anaconda
    c. Click Anaconda Prompt.

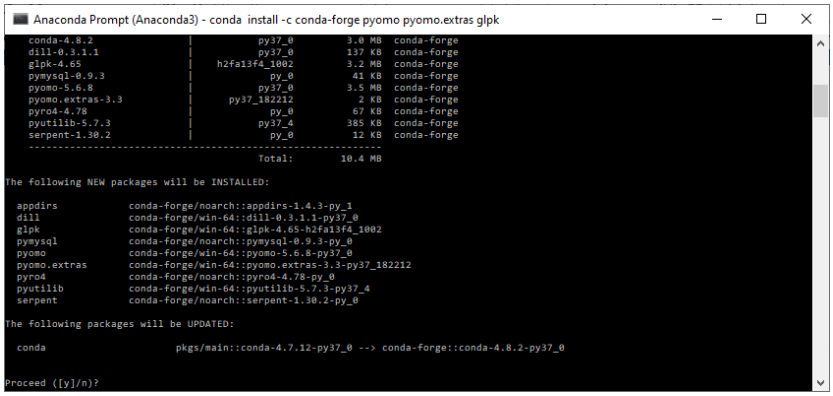
Figure 1: Navigating Anaconda command prompt to install Pyomo

Now you are in the command prompt mode, where you shall type:

conda install –c conda-forge pymo pymos.extras glpk

As seen in the screenshot (Figure 2), entering "y" will complete the installation.


Anaconda Prompt (Anaconda3) - conda install -c conda-forge pyomo pyomo.extras glpk
(base) C:\Users\Mahesh>conda install -c conda-forge pyomo pyomo.extras glpk
Collecting package metadata (current_repodata.json): done
Solving environment: done
==> WARNING: A newer version of conda exists. <==
current version: 4.7.12
latest version: 4.8.2
Please update conda by running
$ conda update -n base -c defaults conda
Package Plan
environment location: C:\Users\Mahesh\Anaconda3
added / updated specs:
- glpk
- pyomo
- pyomo.extras
The following packages will be downloaded:
package | build
appdirs-1.4.3 | py_1 11 KB conda-forge
conda-4.8.2 | py37_0 3.0 MB conda-forge
dill-0.3.1.1 | py37_0 137 KB conda-forge
glpk-4.65 | h2fa13f4_1002 3.2 MB conda-forge
pymysql-0.9.3 | py_0 41 KB conda-forge
pyomo-5.6.8 | py37_0 3.5 MB conda-forge
pyomo.extras-3.3 | py37_182212 2 KB conda-forge
pyro4-4.78 | py_0 67 KB conda-forge
pyutilib-5.7.3 | py37_4 385 KB conda-forge
serpent-1.30.2 | py_0 12 KB conda-forge
Total: 10.4 MB
The following NEW packages will be INSTALLED:
appdirs conda-forge/noarch::appdirs-1.4.3-py_1
dill conda-forge/win-64::dill-0.3.1.1-py37_0
glpk conda-forge/win-64::glpk-4.65-h2fa13f4_1002
pymysql conda-forge/noarch::pymysql-0.9.3-py_0
pyomo conda-forge/win-64::pyomo-5.6.8-py37_0
pyomo.extras conda-forge/win-64::pyomo.extras-3.3-py37_182212
pyro4 conda-forge/noarch::pyro4-4.78-py_0
pyutilib conda-forge/win-64::pyutilib-5.7.3-py37_4
serpent conda-forge/noarch::serpent-1.30.2-py_0
The following packages will be UPDATED:
conda pkgs/main::conda-4.7.12-py37_0 --> conda-forge::conda-4.8.2-py37_0


Anaconda Prompt (Anaconda3) - conda install -c conda-forge pyomo pyomo.extras glpk
conda-4.8.2 | py37_0 3.0 MB conda-forge
dill-0.3.1.1 | py37_0 137 KB conda-forge
glpk-4.65 | h2fa13f4_1002 3.2 MB conda-forge
pymysql-0.9.3 | py_0 41 KB conda-forge
pyomo-5.6.8 | py37_0 3.5 MB conda-forge
pyomo.extras-3.3 | py37_182212 2 KB conda-forge
pyro4-4.78 | py_0 67 KB conda-forge
pyutilib-5.7.3 | py37_4 385 KB conda-forge
serpent-1.30.2 | py_0 12 KB conda-forge
Total: 10.4 MB
The following NEW packages will be INSTALLED:
appdirs conda-forge/noarch::appdirs-1.4.3-py_1
dill conda-forge/win-64::dill-0.3.1.1-py37_0
glpk conda-forge/win-64::glpk-4.65-h2fa13f4_1002
pymysql conda-forge/noarch::pymysql-0.9.3-py_0
pyomo conda-forge/win-64::pyomo-5.6.8-py37_0
pyomo.extras conda-forge/win-64::pyomo.extras-3.3-py37_182212
pyro4 conda-forge/noarch::pyro4-4.78-py_0
pyutilib conda-forge/win-64::pyutilib-5.7.3-py37_4
serpent conda-forge/noarch::serpent-1.30.2-py_0
The following packages will be UPDATED:
conda pkgs/main::conda-4.7.12-py37_0 --> conda-forge::conda-4.8.2-py37_0
Proceed ([y]/n)?

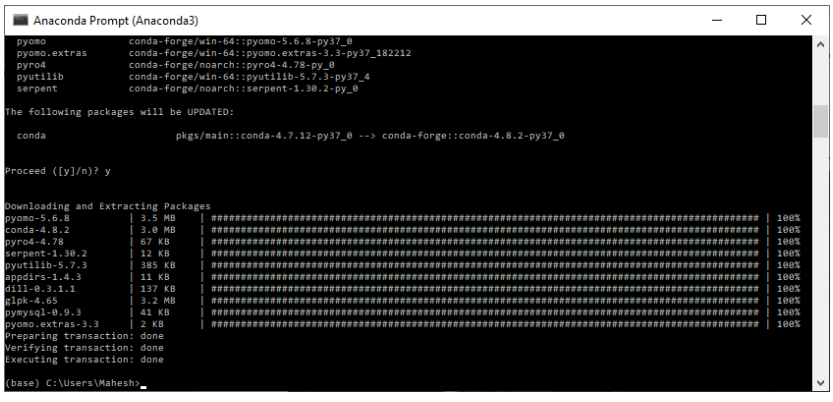

Anaconda Prompt (Anaconda3)
pyomo conda-forge/win-64::pyomo-5.6.8-py37_0
pyomo.extras conda-forge/win-64::pyomo.extras-3.3-py37_182212
pyro4 conda-forge/noarch::pyro4-4.78-py_0
pyutilib conda-forge/win-64::pyutilib-5.7.3-py37_4
serpent conda-forge/noarch::serpent-1.30.2-py_0
The following packages will be UPDATED:
conda pkgs/main::conda-4.7.12-py37_0 --> conda-forge::conda-4.8.2-py37_0
Proceed ([y]/n)? y
Downloading and Extracting Packages
pyomo-5.6.8 | 3.5 MB | ########## | 100%
conda-4.8.2 | 3.0 MB | ########## | 100%
pyro4-4.78 | 67 KB | ########## | 100%
serpent-1.30.2 | 12 KB | ########## | 100%
pyutilib-5.7.3 | 385 KB | ########## | 100%
appdirs-1.4.3 | 11 KB | ########## | 100%
dill-0.3.1.1 | 137 KB | ########## | 100%
glpk-4.65 | 3.2 MB | ########## | 100%
pymysql-0.9.3 | 41 KB | ########## | 100%
pyomo.extras-3.3 | 2 KB | ########## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) C:\Users\Mahesh>

Figure 2: Command prompt scenario while installing Pyomo solver

Once both Anaconda and Pyomo installed in the machine, navigate "Spider" either through the Start button or windows explorer. The graphical user interface of the Spider will be as included in Figure 3.

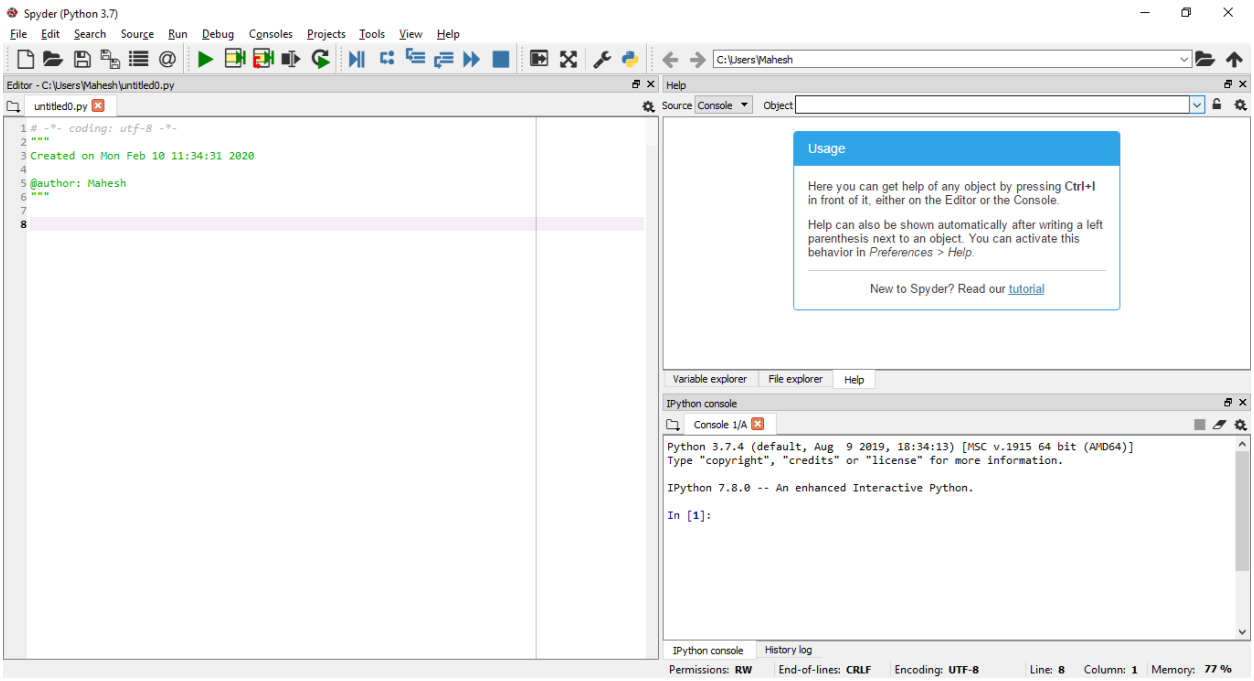


Figure 3: Spider interface, ready for code Python scripts

Before working, please download the materials: code and date from the GitHub repository: https://github.com/mlmaskey/UC-Merced. By typing this address in any browser, you will arrive at the interface (Figure 4) from which you download the required logistics.

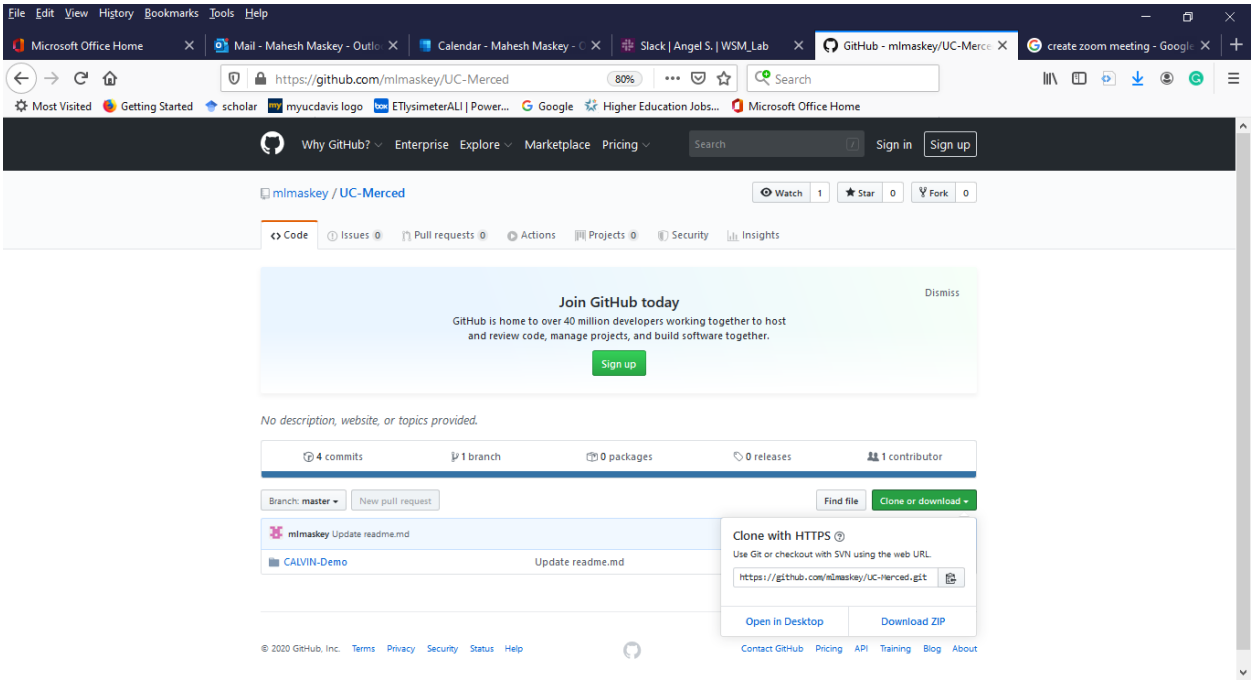


Figure 4: GitHub repository interface to download the CALVIN material.

After downloading UC-Merced-master.zip in your machine where all downloads are stored, move the working folder say "WorkPython." If the folder does not exist, you should create at your desired location where you unzip all the files. Running the CALVIN model requires four input files: (a) classification of CALVIN nodes based on land use and region, "demand_node.csv"; (b) classification of links based on land use, supply basin, region and Central Valley Production model, "portfolio.csv," (c) statistics of all reservoir storages, "SR_stats.csv"; and (d) main CALVIN input file with node, links, amplitude, and costs, links, "link*.CSV" For the better efficient, have the folder as shown in Figure 5 and Figure 6 on a single year and multiple years of runs, respectively. As seen in these figures, the folder "calvin" contains scripts related to CALVIN. The primary folder included main scripts to run the model.

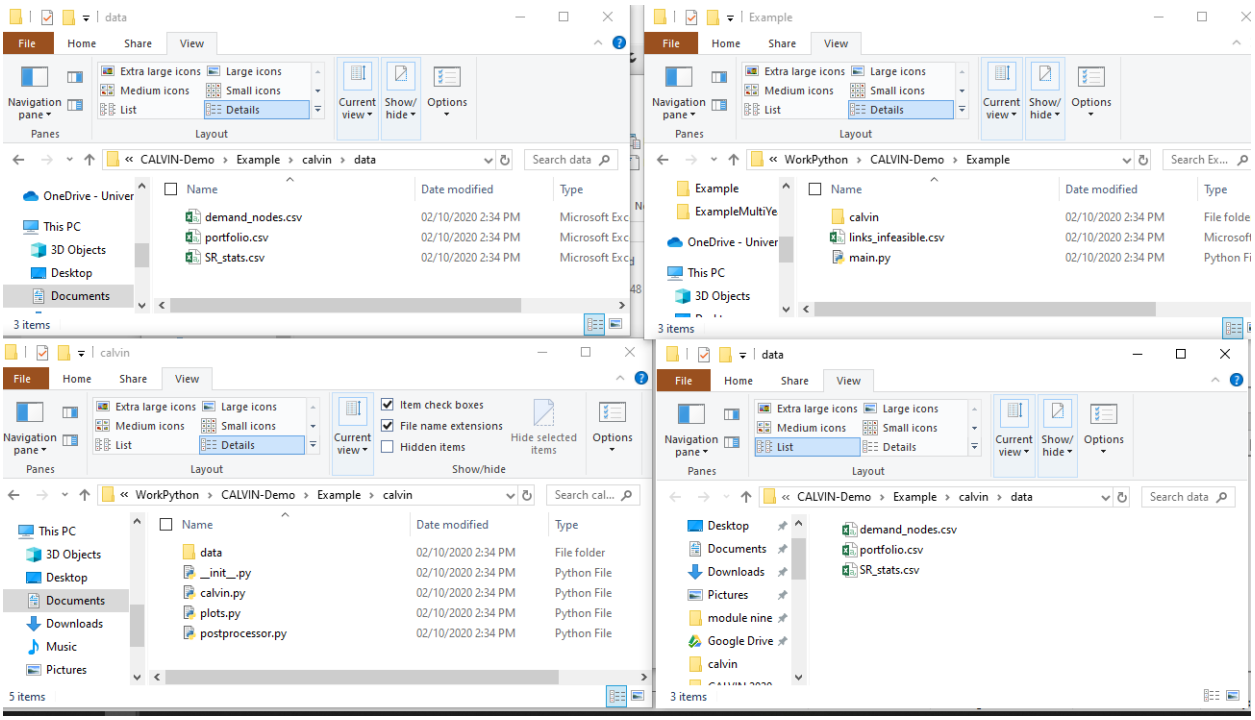


Figure 5: CALVIN Demo Folder for single runs

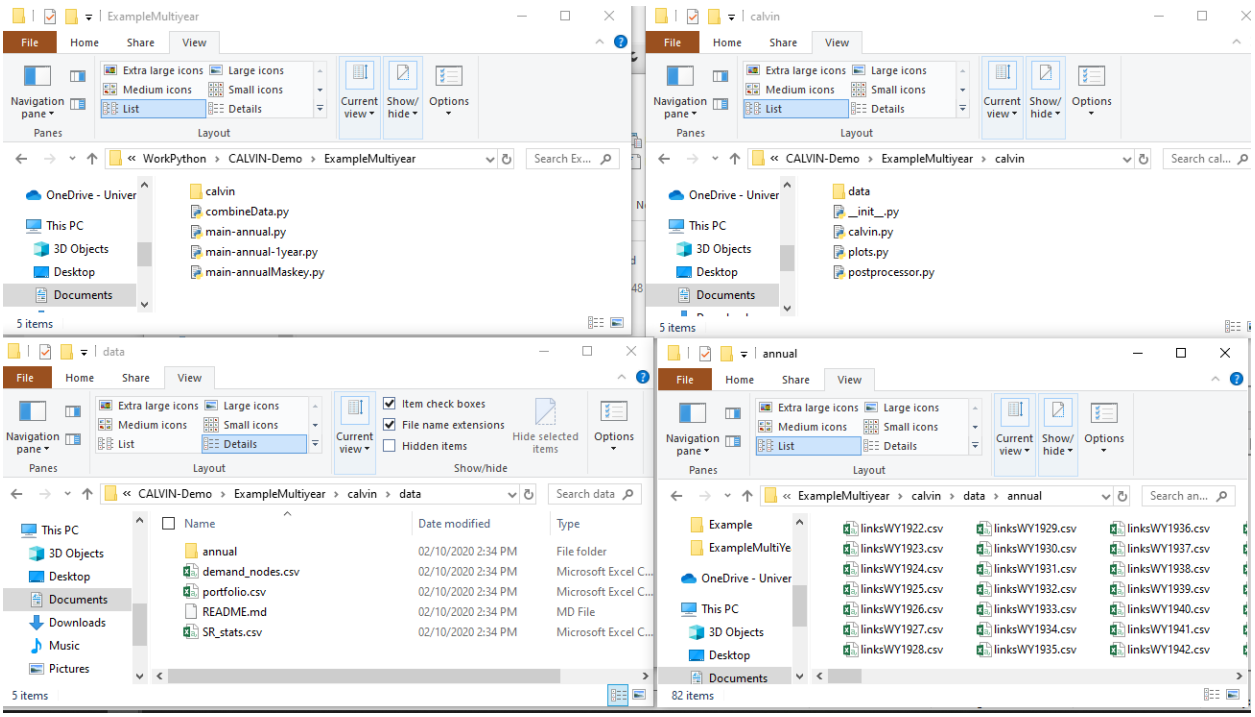


Figure 6: CALVIN Folder for multi-year runs

2.0 Running the example

1. Click the open icon in the Spider interface or go through the File menu.
2. Navigate the "Example" folder.
3. Select the file named "main.py" and click the Open button.
4. Before running the script, let's understand the structure of the texts.
    a. Python modules can get access to code from another module by importing the file/function using import embedded in lines 1 and 2.
    b. Line 5 calls the class named "CALVIN" for the prescribed input file named 'links_feasible.csv."
    c. Line 8 calls a function named "create_pyomo_model()" to create the Pyomo model. This function also includes required constraints and objective function for the optimization exercise.
    d. Based on constraints and objective function, line 11 asks another routine "solve_pyomo_model()" to solve the problem.
    e. Finally, line 14 generates time series files of CALVIN output inside the prescribed folder like "results" via the function "postprocess()" The output files include shadow values, shortage_cost, shortage_volume, evaporation, flow, and storage.
5. Once you open the file and get familiar with the code via comments with hash #, hit the green run button and wait a while.

3.0 Running the example for multiple years

Follow the procedure mentioned in Section 2.0, except the model folder, is "ExampleMultiyear" under "CALVIN-Demo."

1. Instead of "main.py," select the script named "main-annual.py" under the folder ExampleMultiyear
2. Unlike in the example case, you need to choose the range of years to run. Note that you can change the period of runs between 1922 to 2004. For instance, in this example, we choose from 1990 to 2004. The "for-loop" entails the rest of the codes. So, type "1990, 2004" after the keyword "range."

3. While the rest of the codes are similar to the Example case, only the difference is that we need individual input files for each year and save results from different years under a separate folder named with Year like "WYyyyy." We keep all these folders under a prescribed folder like "result/annual," for instance.
4. Once you complete the run as in the previous section, open the script file "combineData" and run it to merge all the output from different consecutive years and store in a single file under the "result" folder.

4.0 Post-processing

The script, "plotUtilities.py," allows us to visualize the dynamics of inflow, storage and evaporation loss, as shown in Figure 7 - Figure 9, in order and bar chart for water storage cost for both agriculture and urban within different regions in Figure 10.

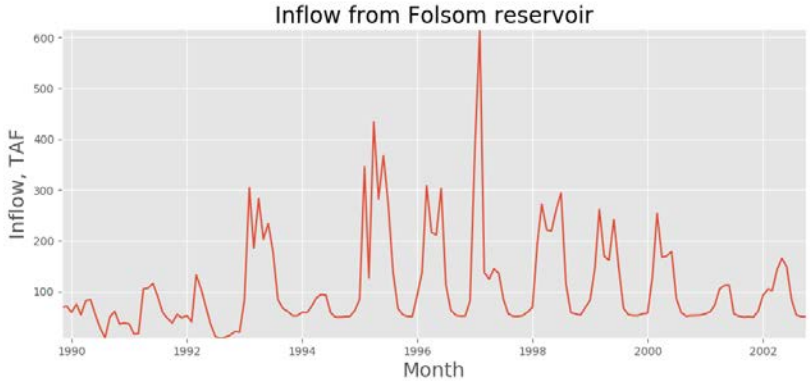


Figure 7: Example plots of the CALVIN output, Inflow from the Folsom reservoir.

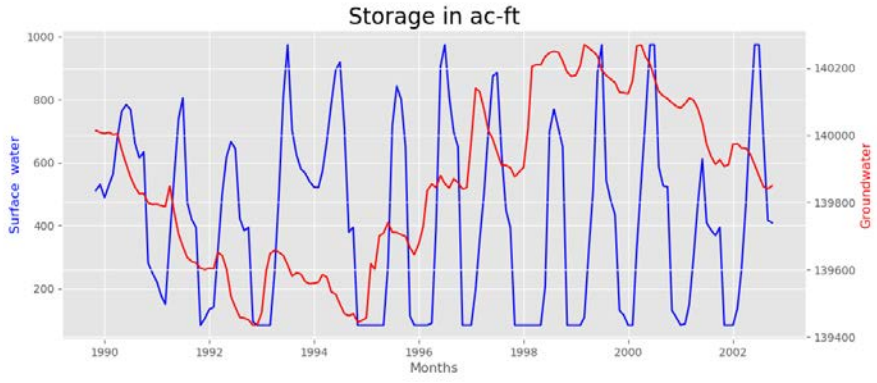


Figure 8: Example plots of the CALVIN output, storage behind the Folsom reservoir.

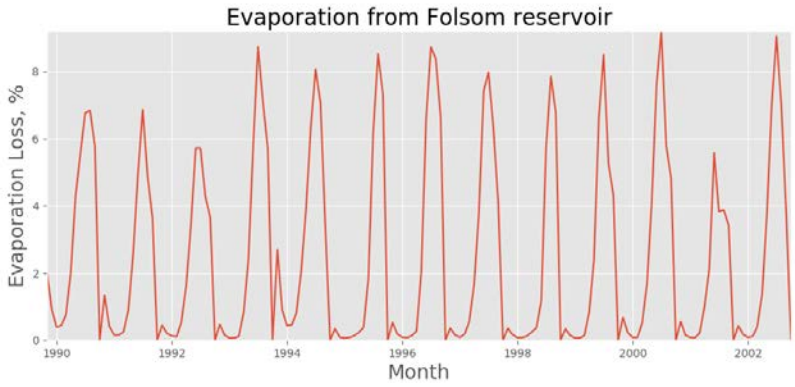


Figure 9: Example plots of the CALVIN output, evaporation loss in percentage from the Folsom reservoir.

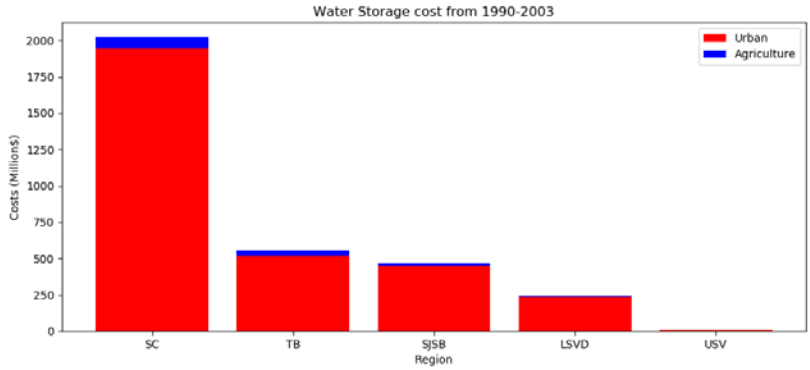


Figure 10: Example plots of the CALVIN output, Water storage cost for the period consider within the system.