Shorter version of workshop tutorial on setting the CALVIN model

By Mahesh Lal Maskey August 5, 2020.

**Table of Contents**

[1.0 Introduction 2](#_Toc47612766)

[2.0 Logistics 2](#_Toc47612767)

[2.1 Installation of Node.js 2](#_Toc47612768)

[2.2 Installation of Git for Windows 3](#_Toc47612769)

[2.3 Installation of Github desktop (optional) 3](#_Toc47612770)

[3.0 CALVIN-network-tool 3](#_Toc47612771)

[3.1 Installing the CALVIN-network-tool 4](#_Toc47612772)

[4.0 CALVIN-Network data overview 6](#_Toc47612773)

[4.1 Data structure 6](#_Toc47612774)

[4.2 Cloning or downloading the CALVIN database 9](#_Toc47612775)

[4.3 Cloning the CALVIN code 11](#_Toc47612776)

[4.4 Environment set up to export CALVIN network data 12](#_Toc47612777)

[4.5 Exporting CALVIN network data 12](#_Toc47612778)

[4.6 Structure of matrix file 15](#_Toc47612779)

[5.0 Installation and set up 16](#_Toc47612780)

[6.0 CALVIN runs 18](#_Toc47612781)

[6.1 Single year run 18](#_Toc47612782)

[6.2 Multiple year run 18](#_Toc47612783)

[7.0 Post-processing 19](#_Toc47612784)

[Appendix A: Additional information 21](#_Toc47612785)

[References 26](#_Toc47612786)

# Introduction

This document provides a step-by-step procedure to install the CALVIN network tool, clone the CALVIN network data, import matrix file for CALVIN input, and install and run the python version of CALVIN, and post-processors. The software should be installed on personal laptops before the workshop series.

# Logistics

## Installation of Node.js

Node.js is just an asynchronous event-driven JavaScript runtime, which helps us build network applications in the California water network implemented in CALVIN. To run the Python version CALVIN, we need Node.js to prepare the input file, which includes information of nodes, links, cost, and limits in a matrix format. At the CALVIN user level, mostly used commands are *npm install* and *cf matrix*. These commands are explained and implemented in the subsequent sections. In order to start with the Node.js tool, one can download the suitable version of Node.JS via <https://nodejs.org/en/download/> (Figure 1).

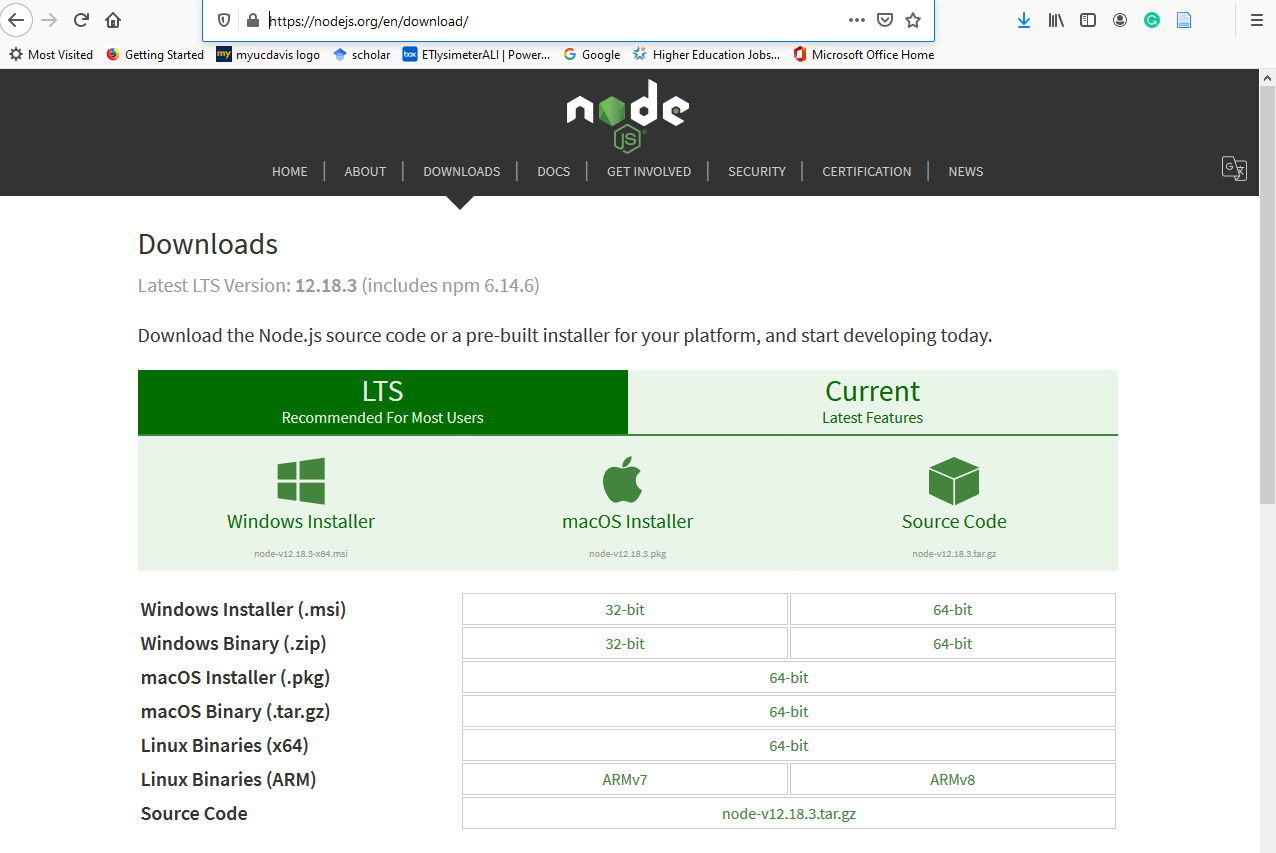


Figure 1: Webpage where Node.js can be downloaded.

Remember that Node.js is necessary to run the CALVIN-network-tool even though it is not widely used; rather, it keeps running in the background.

## Installation of Git for Windows

For windows users, Git is a useful tool to work with commands related to Node.js. Git for windows can be downloaded via <https://gitforwindows.org>. Follow the procedure to install after downloading. To prepare Python readable CALVIN input file, we mostly work on Git using Git Bash interface (Figure 2). Git Bash is simply the command line terminal and similar to the "git" command in LINUX and UNIX environments.

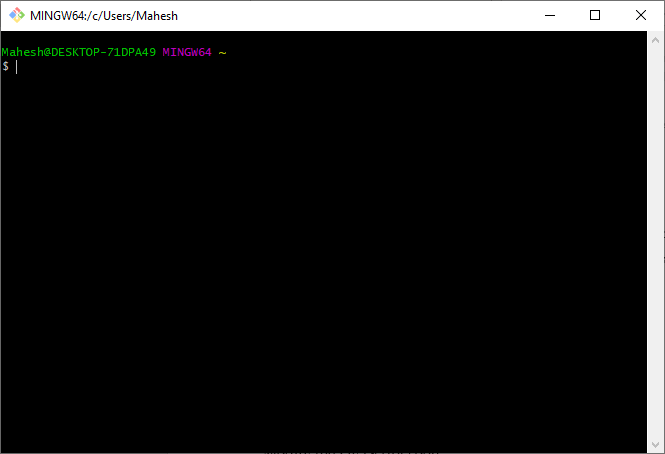


Figure 2: Git Bash interface embedded in Git for Windows.

## Installation of Github desktop (optional)

While the main repository of the CALVIN model is GitHub, one needs to manage codes, data, and network files through the GitHub repository. However, one can manage these files in the local drive, just downloading them without installing GitHub Desktop. Therefore, it is not compulsory to have, but it becomes useful to have it. GitHub Desktop can be downloaded from https://desktop.github.com/. For complete documentation on GitHub Desktop, please refer to <https://docs.github.com/en/desktop>.

# CALVIN-network-tool

Calvin-network-tool is the main core of the preprocessing input file for the python version of the CALVIN. It uses the *cnf* command line tool implemented in JavaScript, which runs the HEC-PRM model or export as a delimited matrix file. This tool entails some commands that are useful to handle the CALVIN-network data. They are *matrix*, *hec-prm*, *library*, *validate*, *list*, and *apply-changes*. The syntax of this tool is:

*cnf [command] [arguments]*

As mentioned that, we will use the Git Bash Interface form cnf command-line tool.

* *matrix*: created a delimited matrix file to run 3rd party solver.
* *hec-prm*: writes and updates HOBBES Network Format CALVIN data to DSS file. The subcommands associated with hec-prm are: build, run, update-repo, show, and debug. For the CALVIN user, these commands are not much used.
* *library*: run cnf maintenance-related command like init and update.
* validate: validates a CALVIN data is in HOBBES network format to check the errors and number of nodes, links, and regions.
* *list*: lists the path of all nodes/links with names and repository path. To list all nodes/links, use all followed by list.

For getting help in all commands, use the help command using:

*cnf [command] –help*

## Installing the CALVIN-network-tool

* Make sure that Node.js, Git for Windows and GitHub Desktop are already installed in the local drive, as mentioned in Sections 2.1, 2.2, and 2.3.
* Create a folder name CALVIN-network in your local drive using windows explorer.
* Open Git Bash through taskbar (**START -> [PROGRAM] -> Git -> Git Bash**). The Git Bash interface shown in Figure 3 will appear. While using Git Bash, Node.js should be running in the background.

Install the CALVIN-network-tool package into CALVIN-network folder just created using the *npm* command of Node.js tool with *--prefix* argument:

***- npm install -g calvin-network-tools --prefix ~/Documents/CALVIN-network***

Note that **~/Documents/CALVIN-network** means:

**C:\Users\[username]\Documents\CALVIN-network**.

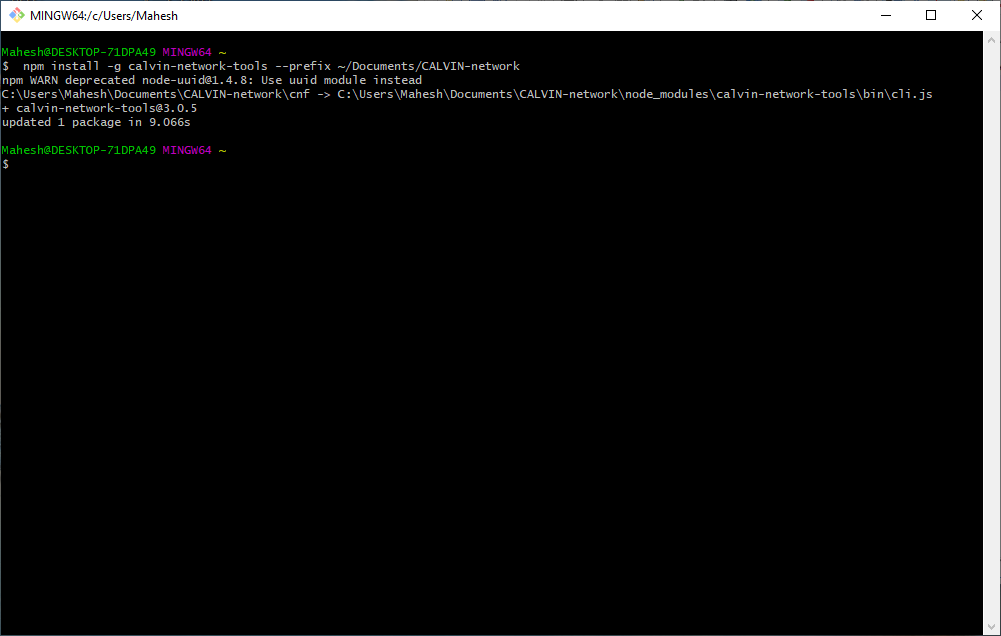


Figure 3: Git Bash interface after installing the calvin-network-tools under CALVIN-network folder

* Add this new location to the git bash path so that it will know where to find the module in this new location. To do this, simply create an empty file named **.bashrc** in the user folder C:\Users\[username]. Then, open this file in whatever text editor (e.g., notepad, notepad++). Add the following line to the file:

***export PATH="/c/Users/[username]/Documents/CALVIN-network":$PATH***

This line tells the Git Bash to search for executables in the local drive before searching in the other typical locations in the $PATH.

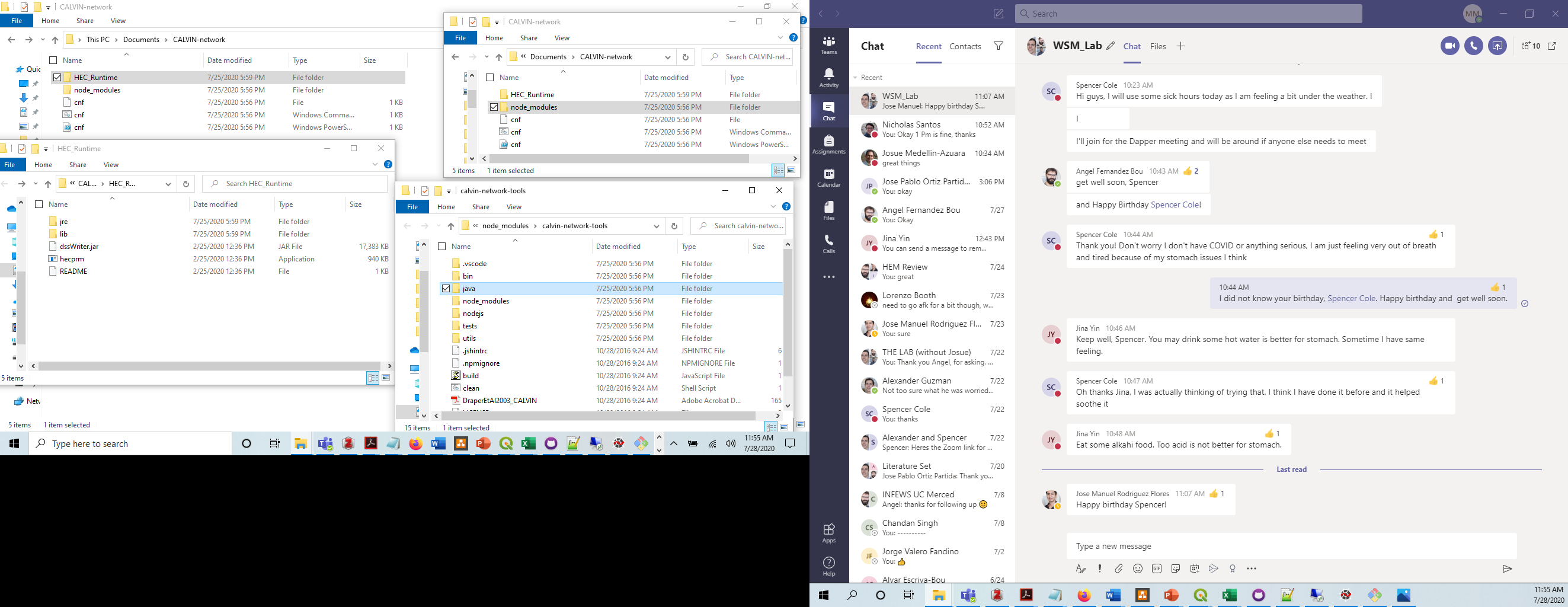


Figure 4: CALVIN network tool folder

* Close and re-open Git Bash
* Once the *calvin-network-tool* package is download, download HEC-runtime (see later) through the GitHub repository[[1]](#footnote-1) and copy or move to CALVIN-network folder. The network tool folder looks like shown in Figure 4.

# CALVIN-Network data overview

## Data structure

CALVIN water network contains hydrology and economic data derived from various sources (Davis and Jenkins 2001; Andrew J. Draper 2000; Howitt et al. 1999). Hydrology implemented in CALVIN was simulated using CALSIM II for the period 1921-2003 (Andrew James Draper 2001). The economic value functions are derived from the Statewide Agricultural Production Model (SWAP) for agricultural demand, relying on demands for the forecasted the year 2020 for the base year 1995 (Howitt et al. 2012; Howitt, Ward, and Msangi 1999, 2001). Besides, urban water demand in the CALVIN represents the projected 2020 population based on information provided by the Department of Water Resources (Jenkins 2000a, 2000b). The cost of other infrastructures was included based on local and regional information collected (Ritzema, Newlin, and van Lienden 2000).

The CALVIN network that represents complex inter-tied water in California is structured in a big database, as shown in Figure 5. Figure 5 shows the major regions of the California water network. The current version of the California network in the CALVIN covers the entire central valley and Southern California. Ten major river basins are included in this network. As listed in Table A 1 and Table A 2, the network encompasses 53 surface water reservoirs and 34 groundwater basins. While 21 groundwater basins are enclosed within the central valley region, and the other 13 basins are in Southern California, as shown in Figure A 1 and Tables A1, A2, and A3. Likewise, there are 35 planning regions, as shown in Figure A 2. While Table A 1 and Table A 2 list the surface water and groundwater reservoirs included in the California water network, Table A 3 presents all the Central Valley regions defined by the Central Valley Production Model (CVPM), including their geographic information.

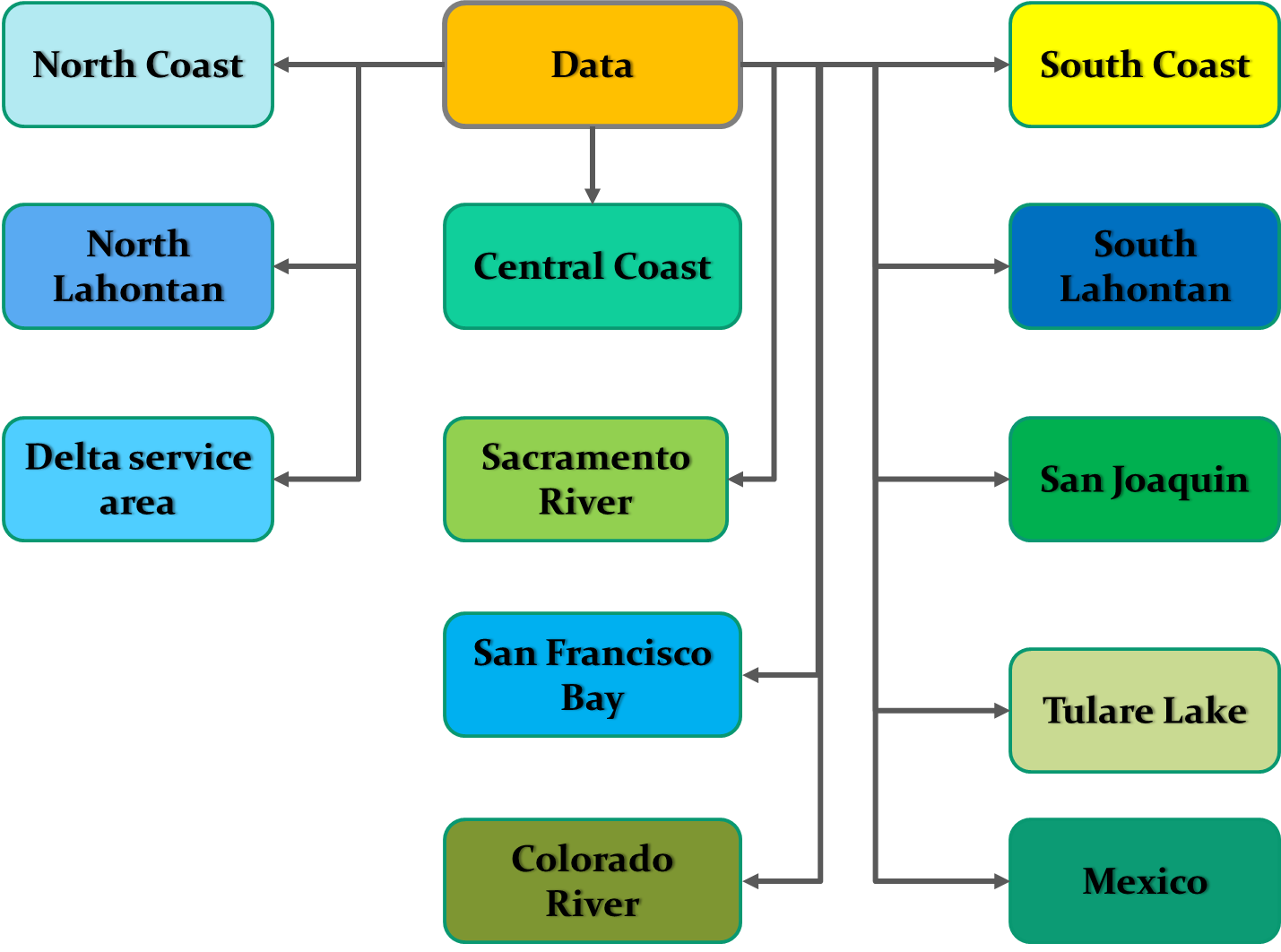


Figure 5: Overall view of CALVIN water network data structure showing major components

The primary folder "data" represents the whole CALVIN network, and subfolders embedded in it are the major regions. Each folder and subfolder encompass water infrastructures and conveyances. The content of the individual subfolder depends on the nature of the facilities. These folders represent either nodes or links included in the California water network. Nodes included in the network represent infrastructures and facilities like reservoirs, demand areas, recharge facilities, sinks while links describe conveyance facilities. These folders include required data related to hydrology, economics, upper (lower) bound, and geo-location information.

Table 1 lists the major files included in the CALVIN network database. All the folders include JSON and GEOJSON files associated with geo-locations of regions, infrastructures, and connection of conveyance facilities. These files are coded in JavaScript language and are useful for detecting the connection between nodes and mapping the network using spatial packages.

Table 1: List of major files that are embedded in CALVIN database

|  |  |  |
| --- | --- | --- |
| **Nature of components** | **File Naming** | **Applied to** |
| Flow through conveyance | flow.csv | Links |
| Surface water or ground water storage | storage.csv | Nodes |
| Inflow to the reservoirs | default.csv | Nodes |
| Upper bound time series | UBT.csv | Links |
| Lower bound time series | LBT.csv | Links |
| Upper bound in month | UBM.csv | Links and nodes |
| Lower bound in month | LBM.csv | Links and nodes |
| Penalty functions | [month Name].csv | Links and nodes |
| Properties of conveyance facilities | link.json | Links |
| Location where link starts | origin | Links |
| Location of ending link | terminus | Links |
| Geomtrical dataof polygons | region.geojson | Regions |
| Properties of infrastructures | node.geojson | Nodes |

For example, the folder that represents a reservoir contains files associated with storage, inflow, upper or lower bound, and penalty. In some cases, it contains a penalty function as well. Likewise, the folder representing a link include flow, bounds, and geo-location.

The nodes in the CALVIN model are represented by alpha characters, followed by numbers except for the surface water reservoir. For instance, *sr\_blb* represents the Black Butte Lake in the Sacramento River within Northwest Valley (Table A 1). Likewise, *gw\_20* is one of the Central Valley groundwater basins in the Tulare Lake Basin (Table A 2). The demand, junction, and diversion nodes are represented by one alpha character followed by at most three-digit integers such as *d701* representing Delta Mendota Canal diversion to CVPM 10 in San Joaquin Valley.

## Cloning or downloading the CALVIN database

* Like in the installation of network tool Section 3.1, create an empty folder name for your project, e.g., myProject.
* Go to a GitHub repository (Figure 6) named Calvin-network-data via <https://github.com/ucd-cws/calvin-network-data>. This repository contains the data for the CALVIN network. HOBBES interface implemented in CALVIN Water Network[[2]](#footnote-2) allows us to visualize this data in a separate interface.
* Click the "Code" button on the upper right (Figure 7) to clone the repository or download it.
* If GitHub Desktop is already installed, click on "Open in GitHub Desktop." If not, or you prefer to download, click on "Download Zip."

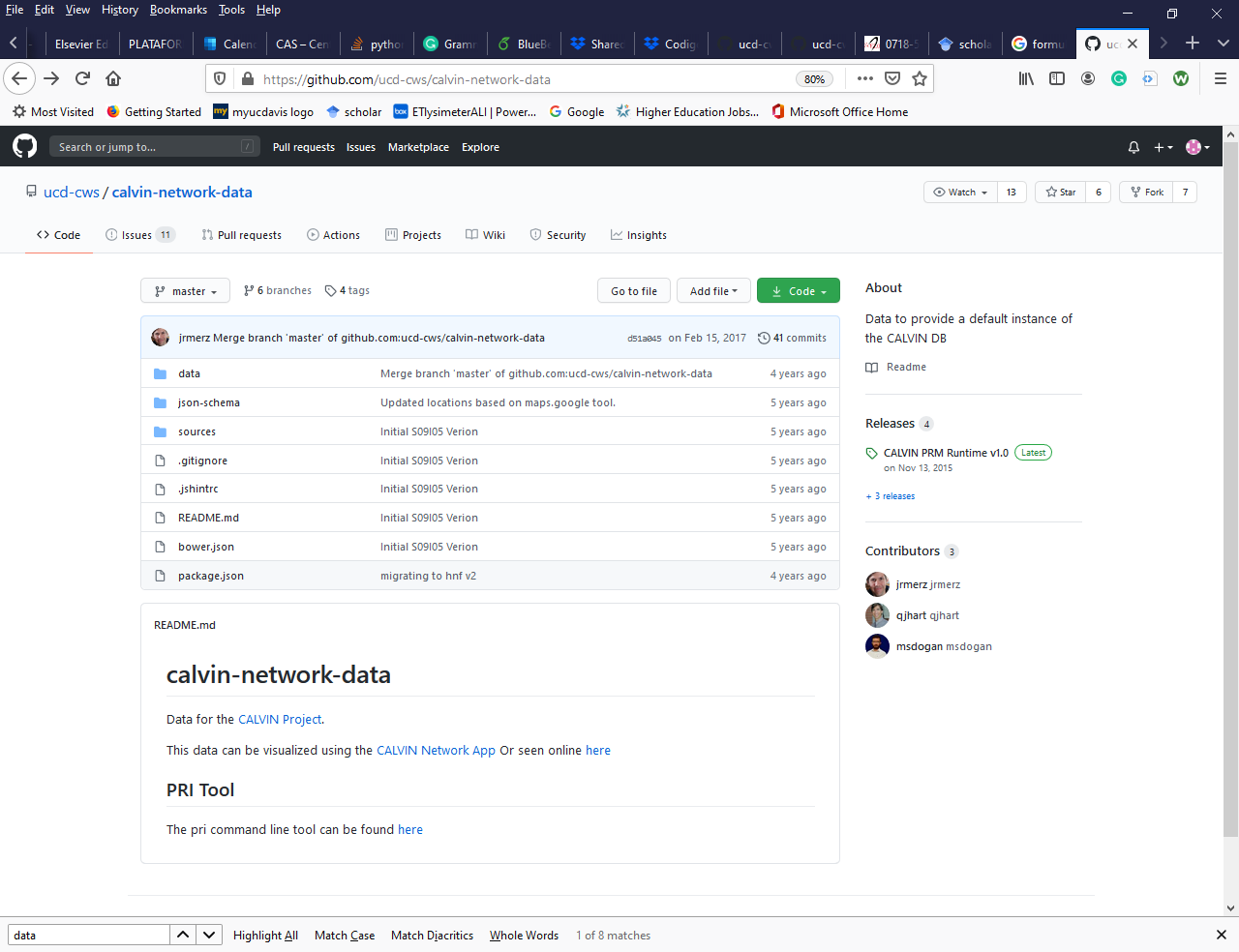


Figure 6: GitHub interface for CALVIN network data.

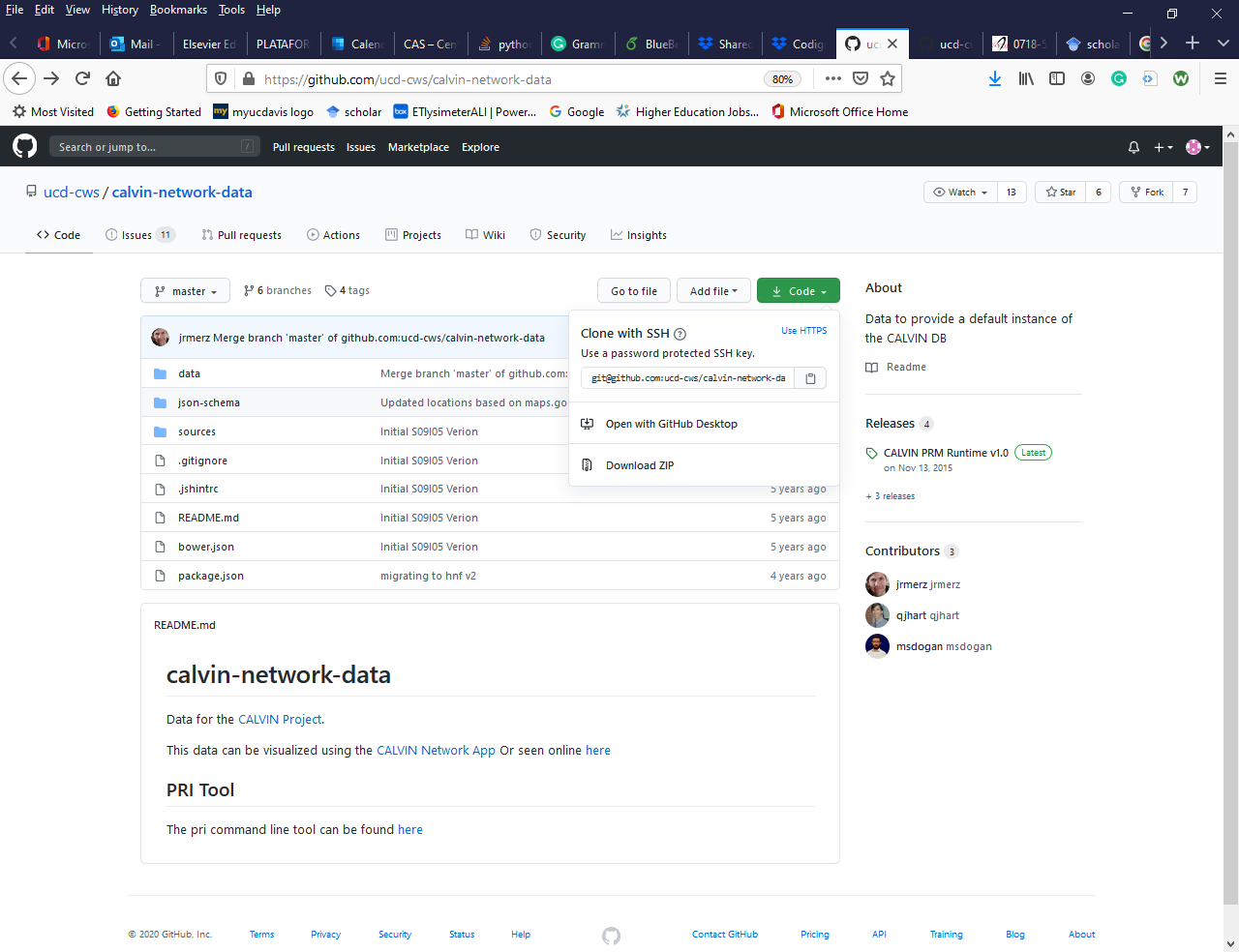


Figure 7: Portion of GitHub interface showing choice of cloning CALVIN network data.

* Once you click Open in GitHub Desktop, it will pop up the Launching GitHub Desktop and launches GitHub Desktop (Figure 8) with Clone a repository dialog box.

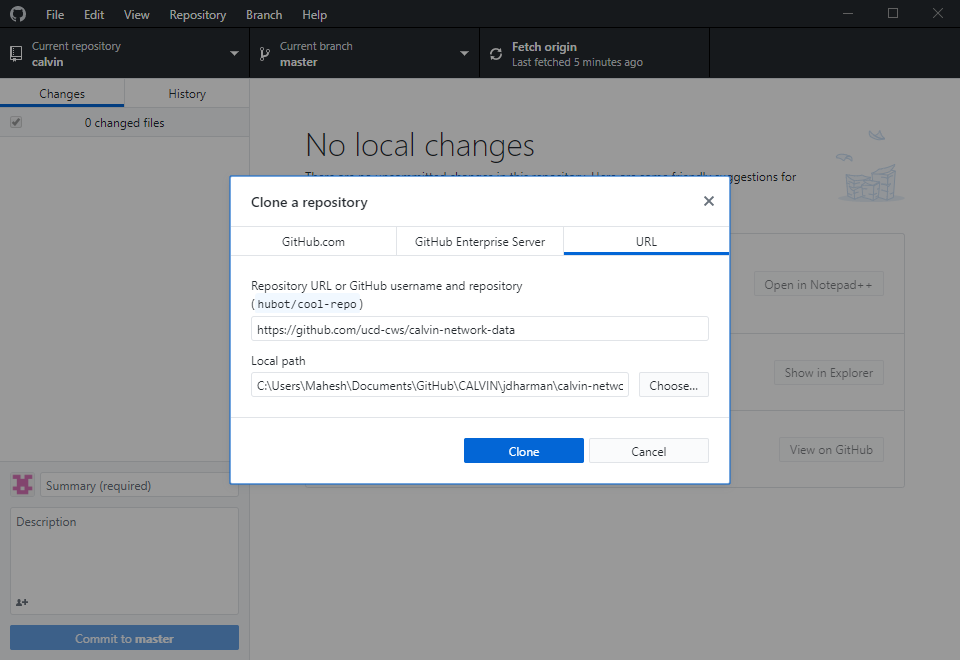


Figure 8: GitHub Desktop interface that allows you to clone that data in desired location

* Click the "Choose" button to select the folder you created. Then, click "Select Folder," You will see a separate interface with a progress message until it completes cloning. Note that the folder where you are cloning the data needs to be empty.
* Alternatively, one can clone the network data in the command line. To work with the command line, do the following:
  + Click the start button and go to Git in the program list.
  + Click GitBash. It will open the GitBaash interface (Figure 2).
  + Then, change the directory to the folder myProject by typing a command like *cd Documents\myProject*.
  + Use the following command: and change the directory to the folder myProject. use the following command:

*$ git clone* [*https://github.com/ucd-cws/calvin-network-data*](https://github.com/ucd-cws/calvin-network-data)

* After completion, you will see a data folder named Calvin-network-data. For simplicity, you can rename as you want, e.g., CALVINdata. It will make use of importing data and post-processing the results in the future.
* If you click Download Zip, the system will download under the default folder. Once download completed, move the zip file to the desired location, unzip it, and manage as mentioned above.

Note that, you may try either one way if one gets failed in cloning.

## Cloning the CALVIN code

* Once again, go to another GitHub repository to clone or download the CALVIN program written in Python. The location of this repository, CALVIN Lessons, is https://github.com/mlmaskey/CALVIN-Lessons.
* Follow the procedure described in 4.2 to clone the code for CALVIN completely.

This repository includes two versions of CALVIN code: (a) single and (b) multiYear. Although most of the files in both folders are common, the run mode is different. The first one, single, allows you to run a single file or single year, independently. The later one, multiYear, runs the code for many years consecutively. The common files in these folders are:

1. “demand\_node.csv” with classes of CALVIN nodes,
2. “portfolio.csv,” a list of link classes based on land use, supply basin, region and Central Valley Production model, and
3. “SR\_stats.csv” for statistics of all surface water reservoir storages

These files are input to the CALVIN runs. The latter is used in the run, and the former two files are mainly used for post-processing.  There is one additional file as the main input to the CALVIN run. The structure of this file is explained later. By default, rs data folder under single includes **infeasible\_links.csv**, which serves as an input for CALVIN run.

## Environment set up to export CALVIN network data

To import CALVIN network data readable by Python version of CALVIN, make sure the folder named "data" exists in the folder where the python code for the CALVIN is. It is also required to make an input file to the python version of CALVIN. This task requires using the "matrix" command mentioned in Section 3.

* Before exporting the CALVIN network data, one needs to prepare a configuration file similar to **.barshc.** For this, create an empty file with a **.prmconf** in any text editor in the user directory.
* Open this file in the text editor and following text:

*{"data":"C:\\Users\\Mahesh\Documents\\myProject\\CALVINData",*

*"runtime":"C:\\Users\\Mahesh\\CALVIN-network\\HEC\_Runtime"}.*

This ".prmconf" file contains two arguments: "data" and "runtime." The former allows the system to search the location of water network data, and later, one helps the command search where the prm tool is. This runtime is required to run the matrix command.

## Exporting CALVIN network data

This section explains the matrix command in detail. The syntax of this command is as follows.

*cnf matrix --format csv --start YYYY-MM--stop YYYY-MM--ts . --to [fileName]--max-ub 1000000000000 --debug all --verbose*

The "matrix" command creates a delimited matrix file to run 3rd party solver. Mostly used arguments are input file type, the period of runs, maximum upper bound, and a couple of keywords. It means the *matrix* command converts the CALVIN network data into a matrix format and writes in a csv file, given by the user, for the specified period. The explanations of later two keywords are not included in this document.

For illustration, let's concentrate on a single-year run. To import the entire network for the entire period, follow the procedures:

* Open Git Bash in your machine.
* Change the directory to data under the "calvin" folder in the Git Bash interface as follow: **cd Documents\myProject\CALVIN-Lessons\single\calvin\data**. Note that this is the folder where you want to import the matrix file.
* Type the command below to import the matrix file "links1922.csv" for the year 1922.

*cnf matrix --format csv --start 1921-10 --stop 1922-10 --ts . --to linksWY1922 --max-ub 1000000000000 --debug all --verbose*

It will take some time to complete the importing matrix. Figure 9 shows the screenshot while running this command. As seen, this screen contains all the arguments required for the matrix command.

It contains the information regarding the period of data, locations of network data, and HEC runtime together with the format of data (Figure 9). Notice that node argument is empty because we are trying to import the entire network for the water year 1921-1922, starting from October 1921 to September 1922. However, the ending period needs to be October 1922 because of the indexing nature of JavaScript. After completion of this command, the file linksWY1922 is stored in the data folder.

Likewise, one can import the partial network with few nodes. For this, one needs to provide values for node argument with node names, as mentioned in Section 2.0. The syntax for this purpose is similar to the previous one but with node argument as below. In this case, one of the arguments *outnodes* is set to nodes, and the end of the above command line follows a list of nodes.

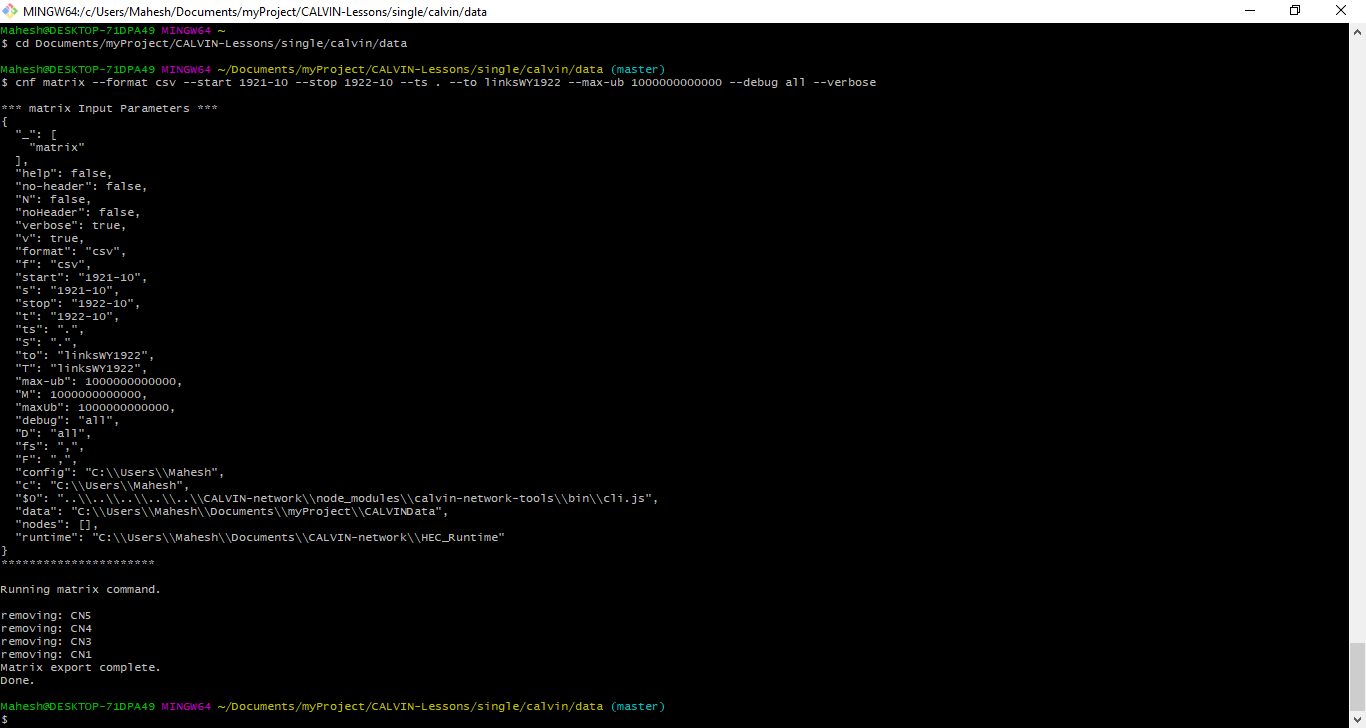


Figure 9: Git Bash interface while running the *matrix* command.

*cnf matrix --verbose --format=csv --start=YYYY-MM --stop=YYYY-MM--ts=. --fs=, --to=[fileName]--outnodes=nodes --max-ub=1000000000 --debug all --verbose [node list]*

For example, to extract the matrx for the link containing nodes *D17*, *SR\_FOL*, and *D9*, the syntax will be:

*cnf matrix --verbose --format=csv --start=2002-10 --stop=2003-09 --ts=. --fs=, --to=linksWY2003Partial --outnodes=nodes --max-ub=1000000000 --debug all --verbose D17 SR\_FOL D9 D85*

In this case, the output file is linksWY2003Partial,csv.

* To import links for multiple years, change the directory to data under the **MultiYear** folder where the full version of the CALVIN code is included.
* Create an empty folder annual inside the data folder because we are going to import the network for multiple periods.

*cd Documents/myProject/CALVIN-Lessons/MultiYear/calvin/data/annual*

* Make sure that the GitHub Bash command line shows the default location. If not, try typing cd, followed by pressing the tab key.
* Make sure the script name "allFiles.sh" has already been downloaded in your machine. If not, download the file through the link: <https://github.com/mlmaskey/CALVIN-Lessons/blob/master/MultiYear/calvin/data/allFiles.sh>. Note that this script contains all the matrix commands for the entire period of simulations from 1921 to 2003. The naming of the output file is **linksWYyyyy.csv**.
* Copy this file to subfolder annual.
* In the command line, type:

*bash allFiles.sh*

This command allows the machine to import multiple files in the prescribed folder. Let the machine runs for a couple of hours.

The procedure, just described, gives the input file for limited foresight runs of the CALVIN. If you need the entire network and the whole period in a single file for the perfect foresight runs, use the syntax like:

*cnf matrix --verbose --format=csv --start=1921-10 --stop=2003-10 --ts=. --fs=, --to=linksPerfect --max-ub=1000000000 --debug all --verbose*

## Structure of matrix file

Figure 10 shows how the matrix input file looks like once you extracted using matrix command. The file includes a header row followed by an alphanumeric matrix. As seen, the header row tells you, the matrix contains columns: *i*, *j*, *k*, *cost*, lower bound, and upper bound. The columns i, j, and k represent an arc originating from node i to terminal node j. Subscript k represents multiple links from origin node i to terminal node j. This component reflects monotonously increasing unit costs represented by the penalty function or the cost curve. Therefore each row represents a link having decision variable. Each variable is associated with the rest three columns: cost , lower bound, , and upper bound, .

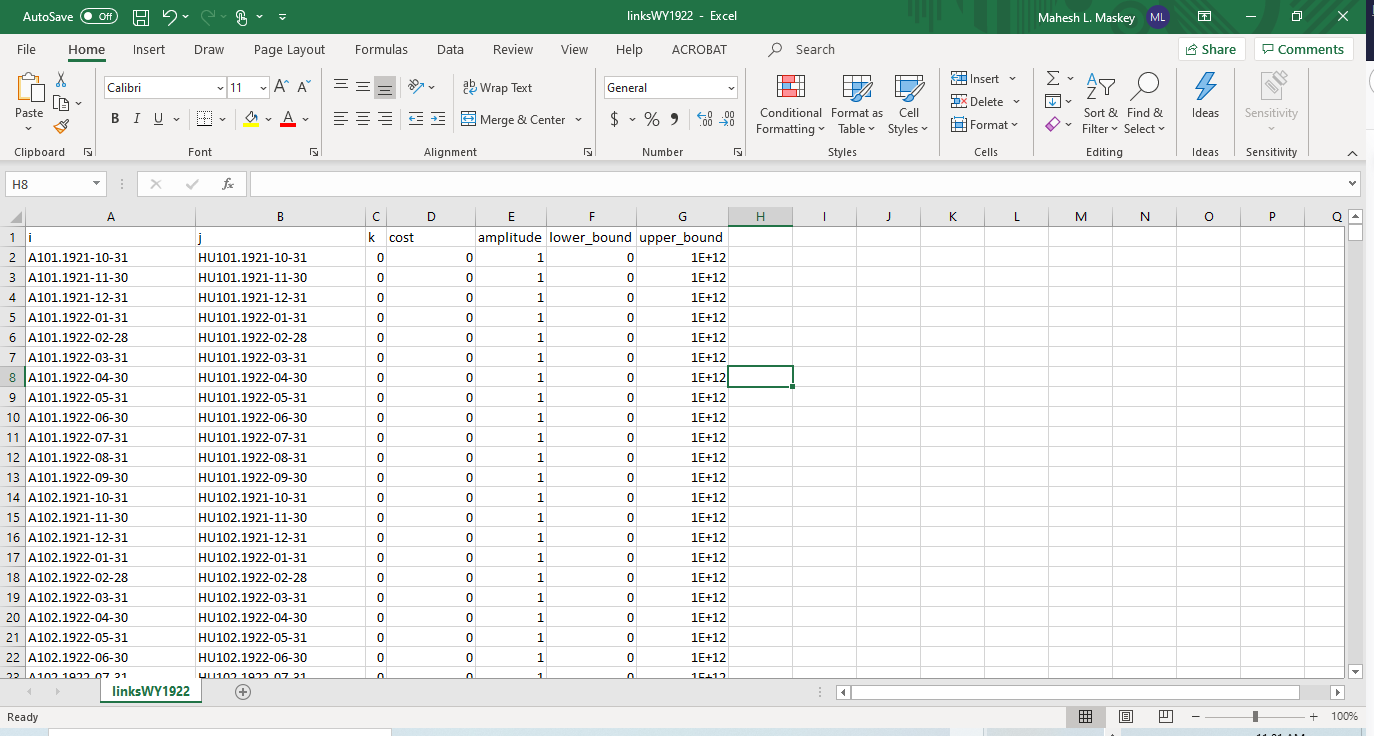


Figure 10: Structure of matrix input file to CALVIN run in Python

# 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.

1. After installing Anaconda, you need to install Pyomo. Window users shall go to the Anaconda command prompt in the following order:
   1. Click Start
   2. Navigate Anaconda
   3. Click Anaconda Prompt.

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 11), entering “y” will complete the installation.

|  |  |
| --- | --- |
|  |  |

Figure 11: 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 12.

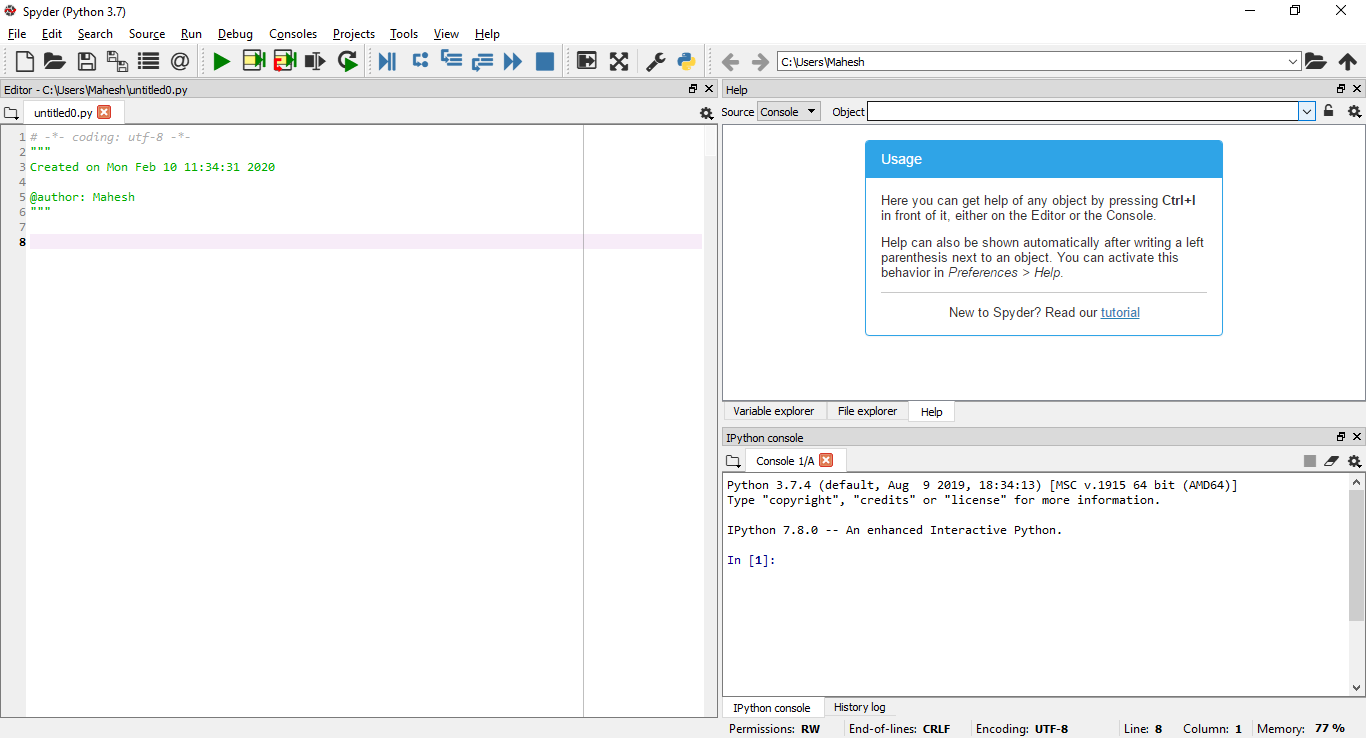


Figure 12: Spider interface, ready for code Python scripts

# CALVIN runs

## Single year run

1. After opening the Anconda Spider interface, click the open button in the toolbar and navigate the "single" folder.
2. Open the script maiin.py.
3. Before running the script, let’s understand the structure of the texts.
   1. Python modules can access code from another module by importing the file/function using import embedded in lines 1 and 2.
   2. Line 5 calls the class named “CALVIN” for the prescribed input file named ‘links\_infeasible.csv.” Note that this input file is a user-defined file with the complete path.
   3. 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.
   4. Based on constraints and objective function, line 11 asks another routine “solve\_pyomo\_model()” to solve the problem.
   5. Finally, line 14 generates time series files of CALVIN output inside the prescribed folder like “results” via the function “post-process().” Like the user-defined input file, the output folder is also a user-defined folder where results from the CALVIN runs are stored in the csv file. They are shadow values, shortage\_cost, shortage\_volume, evaporation, flow, and storage.
4. Once you open the file and get familiar with the code via comments with hash #, hit the green run button and wait a while.

## Multiple year run

1. Instead of the folder "single," navigate the MultiYear folder in the Spider interface.
2. Open the script multi-annual.py under this folder.
3. Here, it would help if you chose 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."
4. The rest of the codes are similar to the single year run. 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.
5. Open the script file named "combineData.py" after completing all the years considered.
6. Then, run it to merge all the output from different consecutive years and store in a single file under the "result" folder.

# Post-processing

The script, “plotUtilities.py,” allows us to visualize the dynamics of inflow, storage and evaporation loss, as shown in Figure 17 - Figure 19, in order and bar chart for water storage cost for both agriculture and urban within different regions in Figure 20.

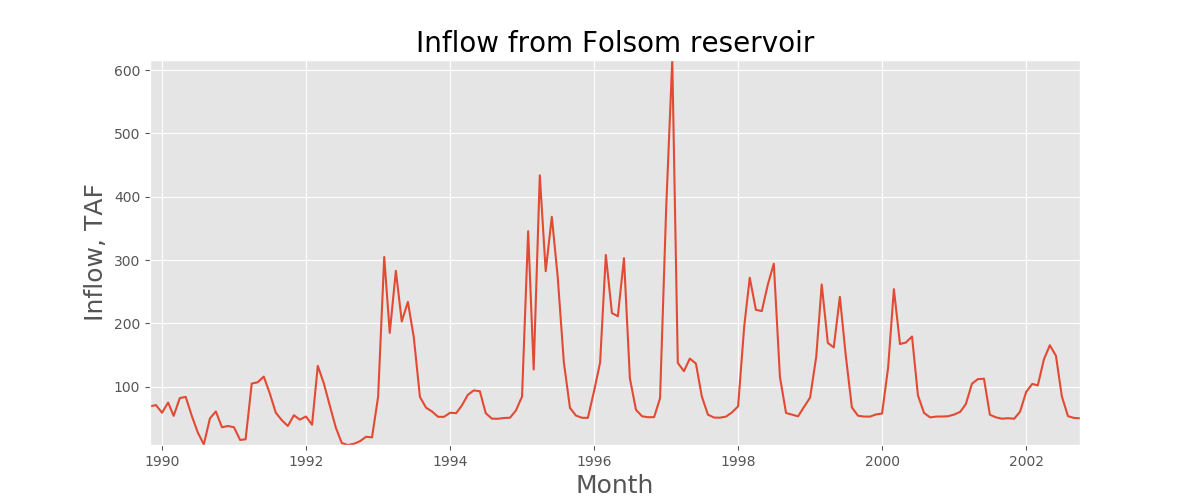


Figure 17: Example plots of the CALVIN output, Inflow from the Folsom reservoir.

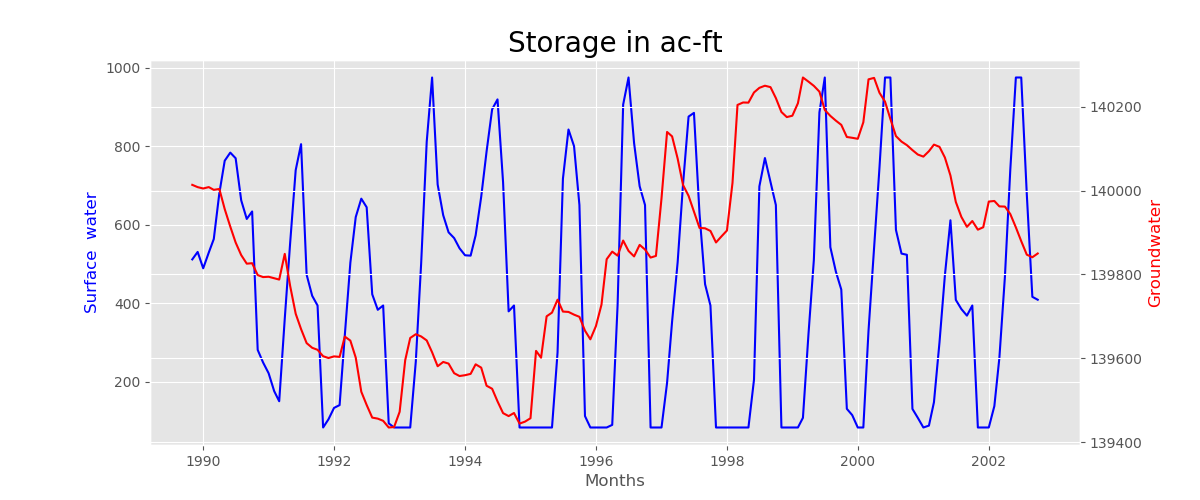


Figure 18: Example plots of the CALVIN output, storage behind the Folsom reservoir.

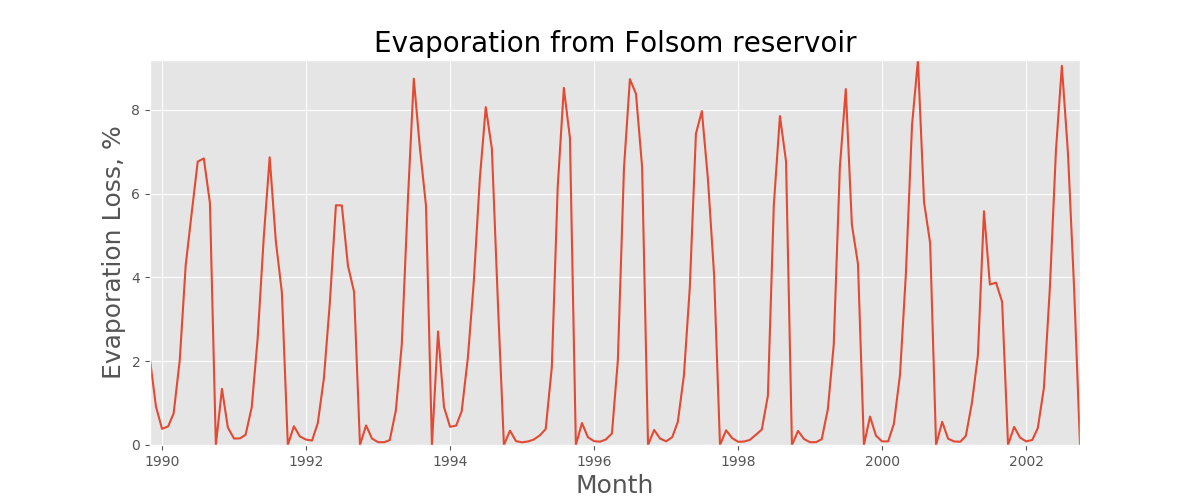


Figure 19: Example plots of the CALVIN output, evaporation loss in percentage from the Folsom reservoir.

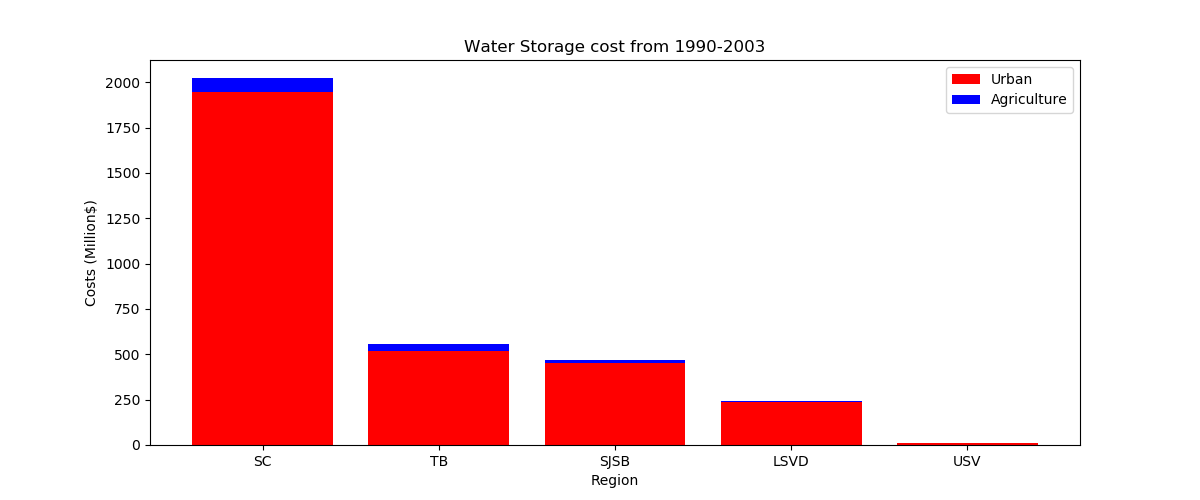


Figure 20: Example plots of the CALVIN output, Water storage cost for the period consider within the system.

# Appendix A: Additional information

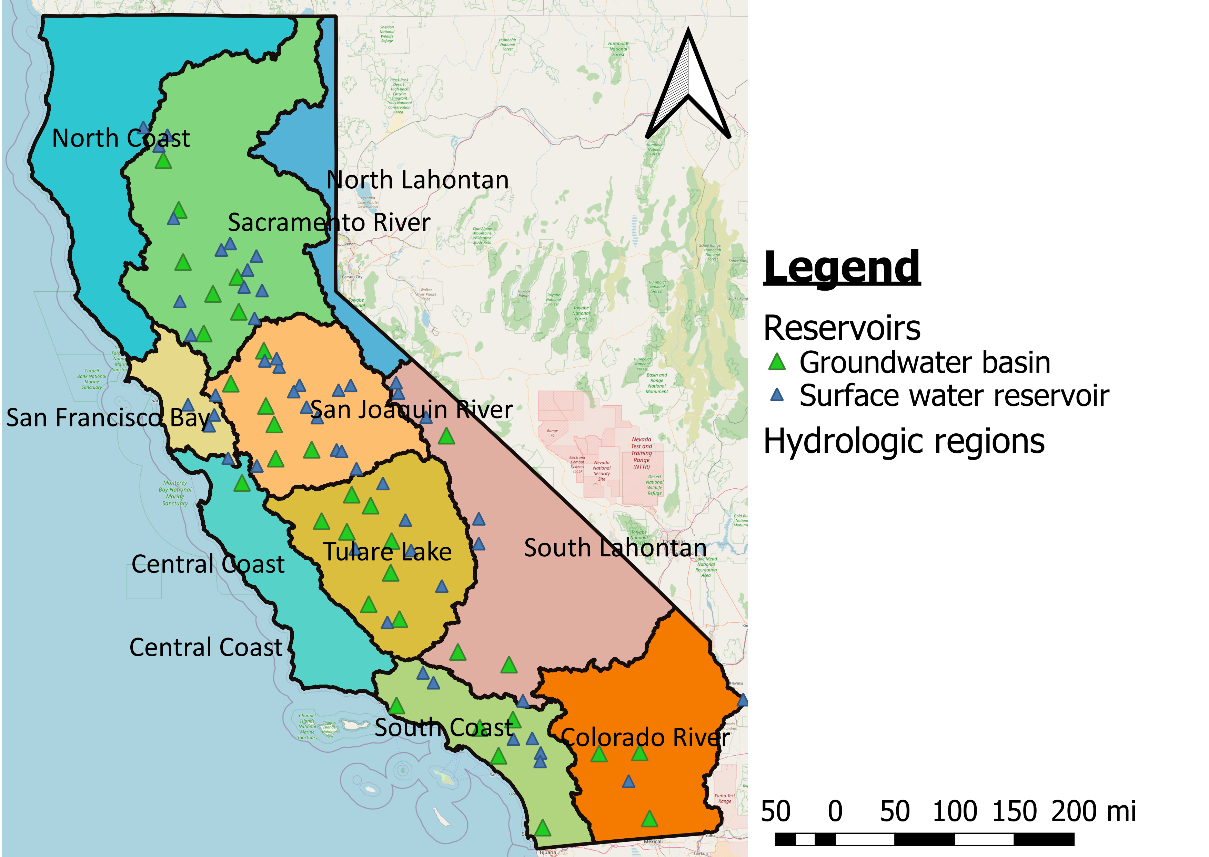


Figure A 1: Hydrologic regions of Caifornia with major reservoirs included in the California water network

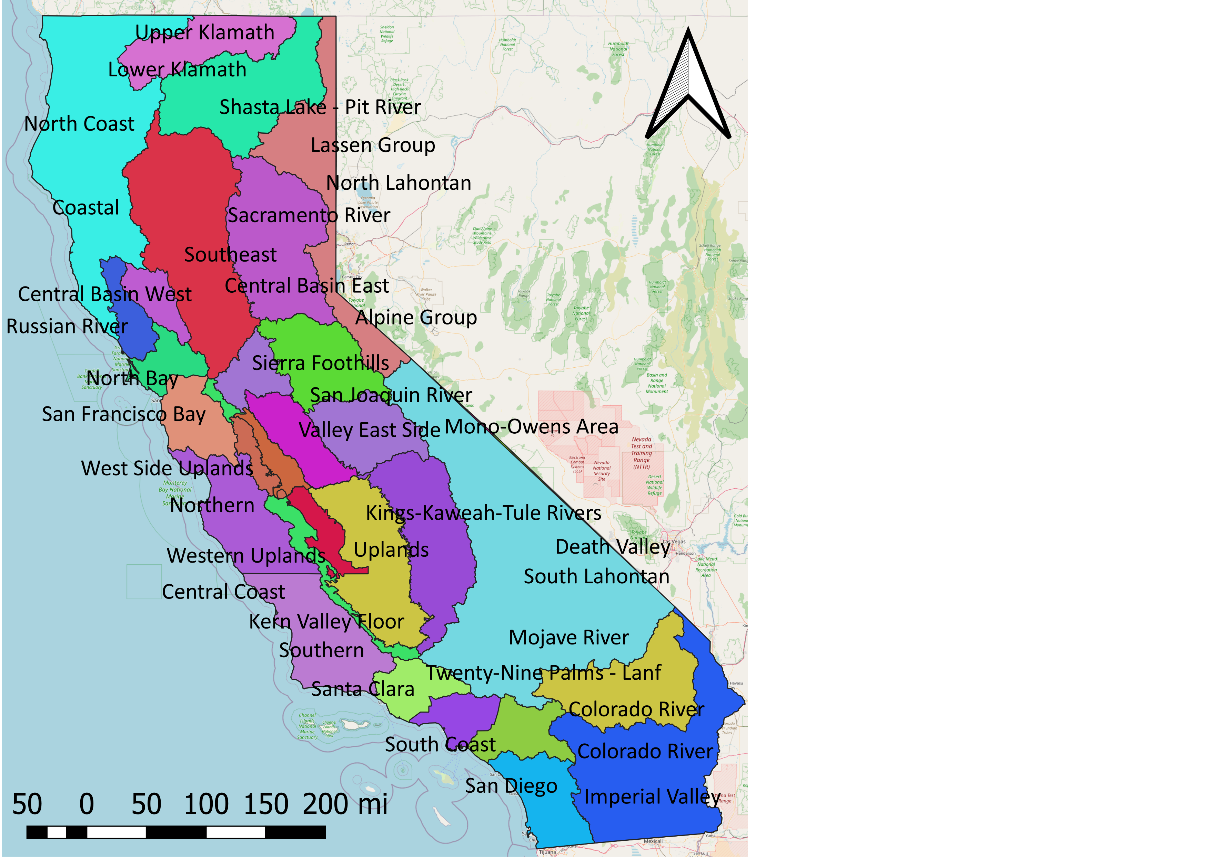


Figure A 2: Planning regions of the state of California

Table A 1: List of surface water reservoirs included in CALVIN water network

| **Surface water reservoir** | **Acronym** | **River basin** | **Region** |
| --- | --- | --- | --- |
| Lake Berryessa | SR\_BER | Sacramento River | Southwest |
| Black Butte Lake | SR\_BLB | Sacramento River | Northwest Valley |
| Eastman Lake | SR\_BUC | San Joaquin River | East Side Uplands |
| New Bullards Bar Res | SR\_BUL | Sacramento River | Southeast |
| Buena Vista Lake Bed | SR\_BVLB | Tulare Lake | Kern Valley Floor |
| Castaic Lake | SR\_CAS | South Coast | Santa Clara |
| Camp Far West Reservoir | SR\_CFW | Sacramento River | Southeast |
| Clair Engle Lake | SR\_CLE | North Coast | Lower Klamath |
| Clear Lake & Indian Valley Reservoir | SR\_CLK\_INV | Sacramento River | Central Basin West |
| Camanche Res | SR\_CMN | San Joaquin River | Eastern Valley Floor |
| Colorado River Storage[[3]](#footnote-3) | SR\_CR1 | Colorado River | Colorado River |
| Colorado River storage[[4]](#footnote-4) | SR\_CR2 | Colorado River | Colorado River |
| Colorado River Storage[[5]](#footnote-5) | SR\_CR3 | Colorado River | Colorado River |
| Lake Crowley[[6]](#footnote-6) | SR\_CRW | South Lahontan | Mono Owens Area |
| Lake Del Valle | SR\_DLV | San Francisco Bay | South Bay |
| Diamond Valley Lake[[7]](#footnote-7) | SR\_DMV | South Coast | San Diego |
| New Don Pedro Reservoir | SR\_DNP | San Joaquin River | Sierra Foothills |
| EBMUD aggregate | SR\_EBMUD | San Francisco Bay | South Bay |
| Englebright Lake | SR\_ENG | Sacramento River | Southeast |
| Folsom Lake | SR\_FOL | Sacramento River | Central Basin East |
| Grant Lake | SR\_GNT | South Lahontan | Mono Owens Area |
| Hensley Lake | SR\_HID | San Joaquin River | East Side Uplands |
| Hetch Hetchy Reservoir | SR\_HTH | San Joaquin River | Sierra Foothills |
| Lake Isabella | SR\_ISB | Tulare Lake | Uplands |
| LAA Storage | SR\_LA | South Lahontan | Indian Wells Area |
| Lake Lloyd/Lake Eleanor | SR\_LL\_ENR | San Joaquin River | Sierra Foothills |
| Los Vaqueros Reservoir | SR\_LVQ | San Joaquin River | Western Uplands |
| Lake McClure | SR\_MCR | San Joaquin River | East Side Uplands |
| Lake Mathews of MWDSC | SR\_MHW | South Coast | Santa Ana |
| Millerton Lake | SR\_MIL | San Joaquin River | East Side Uplands |
| Mono Lake | SR\_ML | South Lahontan | Mono Owens Area |
| New Hogan Lake | SR\_NHG | San Joaquin River | Sierra Foothills |
| New Melones Reservoir | SR\_NML | San Joaquin River | Sierra Foothills |
| Owens Lake | SR\_OL | South Lahontan | Mono Owens Area |
| Lake Oroville | SR\_ORO | Sacramento River | Southeast |
| Pardee Reservoir | SR\_PAR | San Joaquin River | Sierra Foothills |
| Pine Flat Reservoir | SR\_PNF | Tulare Lake | Uplands |
| Lake Perris | SR\_PRR | South Coast | Santa Ana |
| Pyramid Lake | SR\_PYM | South Coast | Santa Clara |
| Rollins Reservoir and Lake Combie | SR\_RLL\_CMB | Sacramento River | Southeast |
| Santa Clara Aggregate | SR\_SCAGG | San Francisco Bay | South Bay |
| Lake Success | SR\_SCC | Tulare Lake | Uplands |
| SF aggregate | SR\_SFAGG | San Francisco Bay | South Bay |
| Shasta Lake | SR\_SHA | Sacramento River | Northeast Valley |
| Lake Skinner | SR\_SKN | South Coast | San Diego |
| Silverwood Lake | SR\_SLW | South Lahontan | Mojave River |
| San Luis Reservoir | SR\_SNL | San Joaquin River | West Side Uplands |
| Salton Sea | SR\_SS | Colorado River | Imperial Valley |
| Thermalito Afterbay | SR\_TAB | Sacramento River | Central Basin East |
| Tulare Lake Bed | SR\_TL | Tulare Lake | Kings Kaweah Tule Rivers |
| Lake Kaweah | SR\_TRM | Tulare Lake | Uplands |
| Tulloch Reservoir | SR\_TUL | San Joaquin River | Sierra Foothills |
| Whiskeytown Lake | SR\_WHI | Sacramento River | Northwest Valley |

Table A 2: List of Groundwater basin included in CALVIN water network

|  |  |  |  |
| --- | --- | --- | --- |
| **Groundwater reservoir** | **Acronyms** | **River Basin** | **Region** |
| Central Valley groundwater basin | GW\_01 | Sacramento River | Northwest Valley |
| Central Valley groundwater basin | GW\_02 | Sacramento River | Northwest Valley |
| Central Valley groundwater basin | GW\_03 | Sacramento River | Central Basin West |
| Central Valley groundwater basin | GW\_04 | Sacramento River | Central Basin East |
| Central Valley groundwater basin | GW\_05 | Sacramento River | Central Basin East |
| Central Valley groundwater basin | GW\_06 | Sacramento River | Central Basin West |
| Central Valley groundwater basin | GW\_07 | Sacramento River | Central Basin East |
| Central Valley groundwater basin | GW\_08 | San Joaquin River | Eastern Valley Floor |
| Central Valley groundwater basin | GW\_09 | San Joaquin River | Delta Service Area |
| Central Valley groundwater basin | GW\_10 | San Joaquin River | Valley West Side |
| Central Valley groundwater basin | GW\_11 | San Joaquin River | Valley East Side |
| Central Valley groundwater basin | GW\_12 | San Joaquin River | Valley East Side |
| Central Valley groundwater basin | GW\_13 | San Joaquin River | Valley East Side |
| Central Valley groundwater basin | GW\_14 | Tulare Lake | San Luis West Side |
| Central Valley groundwater basin | GW\_15 | Tulare Lake | Kings Kaweah Tule Rivers |
| Central Valley groundwater basin | GW\_16 | Tulare Lake | Kings Kaweah Tule Rivers |
| Central Valley groundwater basin | GW\_17 | Tulare Lake | Kings Kaweah Tule Rivers |
| Central Valley groundwater basin | GW\_18 | Tulare Lake | Kings Kaweah Tule Rivers |
| Central Valley groundwater basin | GW\_19 | Tulare Lake | Kern Valley Floor |
| Central Valley groundwater basin | GW\_20 | Tulare Lake | Kings Kaweah Tule Rivers |
| Central Valley groundwater basin | GW\_21 | Tulare Lake | Kern Valley Floor |
| Antelope Valley Groundwater | GW\_AV | South Lahontan | Antelope Valley |
| Coachella Valley Groundwater | GW\_CH | Colorado River | Coachella |
| E&W MWD groundwater basin | GW\_EW | South Coast | San Diego |
| Hayfield Storage Basin | GW\_HF | Colorado River | Chuckwalla |
| Imperial Valley Groundwater | GW\_IM | Colorado River | Imperial Valley |
| Kern-Delta, Arvin-Edison, & Semitropic Projects[[8]](#footnote-8) | GW\_KRN | Tulare Lake | Kern Valley Floor |
| Mojave River Valley Groundwater | GW\_MJ | South Lahontan | Mojave River |
| MWD Service Area[[9]](#footnote-9) | GW\_MWD | South Coast | Metropolitan LA |
| Owens Valley | GW\_OW | South Lahontan | Mono Owens Area |
| SBV groundwater | GW\_SBV | South Coast | Santa Ana |
| Santa Clara groundwater basin | GW\_SC | Central Coast | Northern |
| San Diego groundwater | GW\_SD | South Coast | San Diego |
| Ventura County Groundwater | GW\_VC | South Coast | Santa Clara |

Table A 3: CVPM regions implemented in the CALVIN water network

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CVPM** | **River basin** | **Area[[10]](#footnote-10)** | **DAU Name** | **Region** |
| CVPM 01 | Sacramento River | 421 | Redding East | Northeast Valley |
| CVPM 02 | Sacramento River | 759 | Los Molinos | Northeast Valley |
| CVPM 03 | Sacramento River | 913 | Willows-Arbuckle | Central Basin West |
| CVPM 04 | Sacramento River | 350 | Glenn-Knights Landing | Central Basin West |
| CVPM 05 | Sacramento River | 732 | Durham-Sutter | Central Basin East |
| CVPM 06 | Sacramento River | 612 | Lower Cache Creek | Central Basin West |
| CVPM 07 | Sacramento River | 379 | Placer | Central Basin East |
| CVPM 08 | San Joaquin River | 1001 | Lodi | Eastern Valley Floor |
| CVPM 09 | Sacramento River | 680 | Sacramento Delta | Delta Service Area |
| CVPM 10 | San Joaquin River | 707 | West Side | Valley West Side |
| CVPM 11 | San Joaquin River | 392 | Modesto-Oakdale | Valley East Side |
| CVPM 12 | San Joaquin River | 348 | Turlock | Valley East Side |
| CVPM 13 | San Joaquin River | 1059 | Madera-Chowchilla | Valley East Side |
| CVPM 14 | Tulare Lake | 215 | Kettleman Hills | San Luis West Side |
| CVPM 15 | Tulare Lake | 793 | Tulare Lake | Kings-Kaweah-Tule Rivers |
| CVPM 16 | Tulare Lake | 301 | Academy | Kings-Kaweah-Tule Rivers |
| CVPM 17 | Tulare Lake | 363 | Alta | Kings-Kaweah-Tule Rivers |
| CVPM 18 | Tulare Lake | 868 | Kaweah Delta | Kings-Kaweah-Tule Rivers |
| CVPM 19 | Tulare Lake | 532 | Antelope Plain | Kern Valley Floor |
| CVPM 20 | Tulare Lake | 452 | North Kern | Kern Valley Floor |
| CVPM 21 | Tulare Lake | 340 | Kern Delta | Kern Valley Floor |

# References

Davis, Matthew D, and Marion W. Jenkins. 2001. *Appendix J Groundwater Hydrology*. University of California, Davis.

Draper, Andrew J. 2000. *Appendix I: Surface Water Hydrology.* University of California, Davis.

Draper, Andrew James. 2001. *Implicit Stochastic Optimization with Limited Foresight for Reservoir Systems*. University of California, Davis.

Howitt, Richard E et al. 1999. “Integrated Economic-Engineering Analysis of California’s Future Water Supply.” *Project Completion Rep*.

Howitt, Richard E, Josué Medellín-Azuara, Duncan MacEwan, and Jay R Lund. 2012. “Calibrating Disaggregate Economic Models of Agricultural Production and Water Management.” *Environmental Modelling & Software* 38: 244–58.

Howitt, Richard E., Kristin B. Ward, and Siwa M. Msangi. 1999. *Appendix A: Statewide Water and Agricultural Production Model*. University of California, Davis.

Howitt, Richard E, Kristin B Ward, and Siwa M Msangi. 2001. “Statewide Water and Agricultural Production Model.” *Department of Agricultural and Resource Economics, University of California, Davis*.

Jenkins, Marion W. 2000a. *Appendix B1: Representation of Urban Water Demands and Local*. University of California, Davis.

———. 2000b. *Appendix B2: Urban Value Function Documentation*. University of California, Davis.

Ritzema, Randall S., Brad D. Newlin, and Brian J. van Lienden. 2000. *Appendix H: Infrastructure*. University of California, Davis.

1. https://github.com/ucd-cws/calvin-network-tools/releases/download/v1.4/HEC\_Runtime.zip [↑](#footnote-ref-1)
2. <https://cwn.casil.ucdavis.edu/> [↑](#footnote-ref-2)
3. First priority Colorado River 4.4 water for Policy 4a (661.43 taf/yr) [↑](#footnote-ref-3)
4. Second priority Colorado River 4.4 water for Policy 4a (3.55 maf/yr) [↑](#footnote-ref-4)
5. Second priority Colorado River 4.4 water for Policy 4a (3.55 maf/yr) [↑](#footnote-ref-5)
6. It is also called as Long Valley Reservoir [↑](#footnote-ref-6)
7. Formerly it is known as Eastside reservoir [↑](#footnote-ref-7)
8. Extra groundwater storage - Kern-Delta, Arvin-Edison, & Semitropic Projects [↑](#footnote-ref-8)
9. Additional empty GW Capacity in MWD Service Area [↑](#footnote-ref-9)
10. Area in 1000 acres [↑](#footnote-ref-10)