



**Sandia
National
Laboratories**

Documentation of Customizable Python Scripts for Automating Feeder Reconfigurations and Analyses in CYME

Joseph A. Azzolini, Logan Blakely, Matthew J. Reno

September 2024

SAND2024-123330



Sandia National Laboratories is a multission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview

This file provides the documentation for several Python scripts that were developed by Sandia National Laboratories to enhance the automation and customization capabilities for performing various distribution system planning and analysis tasks in CYME. Specifically, these scripts (.py files detailed in Figure 1) enable the user to evaluate different distribution system configurations and the resulting impacts on hosting capacity results and other metrics. In general, these scripts—and the accompanying documentation—provide the foundation upon which future customized tools can be created. For example, the scripts show how to extract and modify parameters of various circuit components, set up and run analyses using built-in CYME tools (iteratively), and export reports for further evaluations and comparisons. Thus, the capabilities and syntaxes used in the scripts can be adapted and leveraged for countless other objectives.

Note: The user should start with the ‘SetSwitches_Script.py’. The ‘SetSwitches_Script.py’ can be run immediately with the NetwConfOptimiz.sxst’ tutorial study included with the Network Configuration Optimization module. The ‘SetSwitchRunDrive_Script.py’ can also be run with the NetwConfOptimiz.sxst model as long as the user has the EPRI DRIVE module installed with a valid license. The other scripts require a study file to be provided by the user.

CymPy is available as an add-on package available for CYME that provides ways to access the electrical model in CYME, manipulate equipment on the network, and generate reports along with functions, classes, and modules for automating workflows. This document explains how to set up a standalone python environment to interface with the CymPy package that is utilized in the custom automation scripts. Descriptions of each script are outlined below, including details about their inputs, outputs, and features. Additional discussion is provided regarding how the scripts can be modified for other system planning and analysis tasks. This document also contains links, resources, and tips for working with CymPy when adapting the scripts for other purposes.

CymPy is accessible for use within the CYME software (i.e., embedded mode) and through any Python development environment (i.e., stand-alone mode). The scripts described in this document were developed using the stand-alone option due to the ease of integrating third-party Python packages. Note that this document was written in reference to CYME 9.4 Rev 01, Network Reconfiguration Optimization Module (CYME 9.4), EPRI DRIVE v4.0, and Python 3.11.4, and that the computer running the python environment must have access to the CYME software licenses, including the “CYME Scripting Tool with Python” module.

A summary of all the scripts and other supporting files being shared is shown in Figure 1.









Name	Date modified	Type	Size
 cympy.pth	9/9/2024 10:55 AM	PTH File	1 KB
 CymPy_ScriptDocumentation_v2.docx	9/13/2024 9:24 AM	Microsoft Word Docum...	1,281 KB
 environment.yml	9/13/2024 9:38 AM	YML File	16 KB
 MultipleNCO_ExampleScript.py	9/9/2024 10:14 AM	PY File	23 KB
 SetSwitches_Script.py	9/13/2024 10:02 AM	PY File	14 KB
 SetSwitchesRunDrive_Script.py	9/13/2024 10:03 AM	PY File	19 KB
 SingleNCO_ExampleScript.py	9/9/2024 10:12 AM	PY File	23 KB
 SwitchingDeviceStates_Manual_NCO.csv	9/13/2024 9:24 AM	Microsoft Excel Comma...	2 KB

Figure 1 - Summary of Custom Python Scripts and Supporting Files

Python Environment

The automation scripts utilize several third-party Python packages. The provided **environment.yml** file can be used to create a new “environment” in Python 3.11.4 that includes all the required dependencies for running the custom scripts. Two examples for creating this new environment—named “cympyEnv” here—are provided below:

Example directly in Python: `python -m venv c:\path\to\cympyEnv.yml`

Example with Anaconda Python: `conda env create --name cympyEnv --file=c:\path\to\cympyEnv.yml`

Once the *cympyEnv* has been created, it must be activated before running any of the custom scripts.

Importing the CymPy Package

After creating and activating the *cympyEnv*, the steps below can be used to import the CymPy package:

1. Create a text file called “cympy.pth” that contains the path to the parent folder where the CymPy library is located (e.g., “C:\Program Files\CYME\CYME_9_4_rev01”). An example is provided in the zip folder, but the path must be updated accordingly.
2. Place “cympy.pth” in the “site-packages” folder of the *cympyEnv* environment (e.g., “C:\Users\...\Anaconda3\envs\cympyEnv\Lib\site-packages”)
3. Include “import cympy” at the top of any Python script for which the package will be used.

At this point, CymPy should be accessible within any Python script. If any issues occurred with the CymPy import, errors will be thrown in the Python console. Error messages may appear if the script cannot locate the CymPy package, or if the CYME licenses are not available to that machine (e.g., if all licenses for the CYME Scripting Tool with Python are in use by other machines).

Example Workflow for Automating Feeder Configuration Optimization

Consider the scenario shown in Figure 2. In this case, Feeder 1 already has a high penetration of existing distributed solar, and is approaching its PV hosting capacity limit. However, there are sections of the feeder that have minimal solar installations. In this case it is possible to reconfigure the switches to increase the total hosting capacity of the system. This reconfiguration is shown in Figure 3, where two switches have changed position. This has allowed Feeder 2 to pick up several sections of Feeder 1. Feeder 2 still has plenty of available hosting capacity, and Feeder 1 may also recoup some hosting capacity as two of the PV systems at the end of the feeder were transferred to Feeder 1. Overall, this has allowed the total system hosting capacity to increase.

In this simple example it is clear what the change in switch configuration should be, however in a real system there may be hundreds or thousands of possible switch configurations. This requires an automated solution.

Figure 4 shows an example workflow for evaluating different system configurations. This workflow corresponds to the workflow in the `MultipleNCO_Script.py` file. This script leverages python scripting and several analysis modules in CYME to implement a custom, *automated* solution for evaluating the hosting capacity objectives. The Network Configuration Optimization module can be run to generate a subset of possible switching operations. Large systems may have near-infinite possible permutations of switch states, so optimization methods are required. Alternatively, manually switch states could be provided within that workflow. Then, the script iterates through the valid configurations using the EPRI DRIVE module to evaluate the impacts to hosting capacity or other system metrics. Using different load and generation models, we can also analyze the impacts of seasonality on the results.

While the max system-wide hosting capacity may not change drastically, certain locations or branches can have significant impacts. Overall, this automation scripting can be adapted for many uses such as maximizing the total hosting capacity for a feeder (general investment planning) or increasing the hosting capacity at a particular location because of a large DER interconnections request (interconnection process).

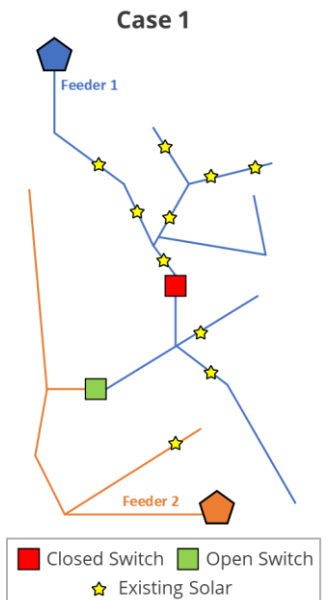


Figure 2 - Example Case 1

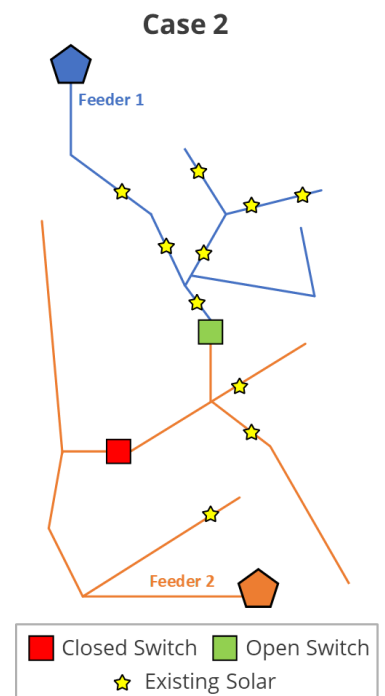


Figure 3 - Example Case 2

CYME Model and Study Setup

CymPy can work with any CYME model and study system. In order to run automatic circuit reconfiguration and hosting capacity though, certain modeling aspects were deemed necessary.

First, all switches must be modeled—including normal open switches between feeders. It is common for distribution systems to be modeled in their normal configuration, in which case modeling the normal open switches between feeders may be unnecessary for many analyses. However, in order to test different circuit configurations and evaluate the impact on various circuit metrics, all switches must be modeled so that their states can be modified to form new configurations.

Second, the presence of electrical loops within the model can cause issues for some analyses. Specifically, the Network Configuration Optimization module in CYME would not run properly if loops existed, which meant that certain parallel conductors had to be merged and substation circuit breaker states had to be modified.

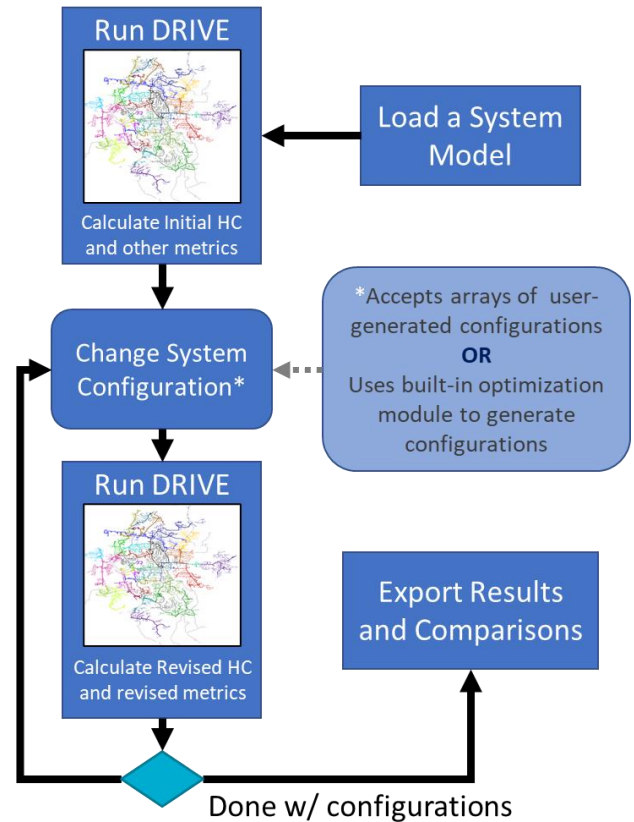
Third, the method by which loads are modeled was found to be important when running hosting capacity analyses with the EPRI DRIVE module. Ideally, a distinct “load model” is created to represent conditions for a certain point in time. For example, when the load model for the summer peak was activated, the actual parameters of all the spot loads in the model would change to their value at that point in time. Then, in the “Load Conditions” settings for DRIVE, the specific load models for peak and minimum loading conditions can be set, providing a more precise representation than using scaling factors that modify all loads equally. Moreover, this scaling factor option is incompatible with load modeling methods that already rely on scaling factors versus actual parameter modifications.

Each of these modeling considerations must be addressed in the desired study file.

For the ‘SetSwitchesScript.py’ and the ‘SetSwitchesRunDrive_Script.py’, the tutorial study included with the Network Configuration Optimization module ‘NetwConfOptimiz.sxst’ can be used without alterations.

Python Script Details

Sandia developed four different scripts that each use CymPy. The following sections provide descriptions of each scripts and detail their specific inputs and outputs.



SingleNCO_ExampleScript.py

The SingleNCO_ExampleScript.py calculates the hosting capacity using EPRI DRIVE, runs a single Network Configuration Optimization (NCO) using the default settings, and then runs EPRI DRIVE again to see how the hosting capacity has changed.

Inputs:

- CYME .sxst study file

Outputs:

- SwitchingDevicesStates_Initial.csv - A CSV file containing the initial states of all switches, reclosers, and breakers in the study, as well as device ID's and device types.
- HCReport_Initial.xlsx - Excel file containing the initial hosting capacity results. This is the same information as in the 'Hosting Capacity Summary Report' obtained through the CYME GUI
- OptReport.xlsx - Excel file containing the results from the Network Configuration Optimization tool. This is the same information as in the 'Network Configuration Optimization – Summary' report obtained through the CYME GUI.
- SwitchingDeviceStates_AfterOpt.csv - CSV file containing the states of all switches with the changes applied from the Network Configuration Optimization tool, as well as device ID's and device types.
- SwitchingStates_BeforeAfter.csv - CSV file containing both the initial and post-optimization states of all switches, reclosers, and breakers in the system, as well as device ID's and device types.
- HCReport_AfterOpt.xlsx - Excel file with the hosting capacity results with the new switching device configuration suggested by the Network Configuration Optimization tool. This is the same information as in the 'Hosting Capacity Summary Report' report obtained through the CYME GUI.

Figure 4 - Sample Workflow with EPRI Drive

MultipleNCO_ExampleScript.py

The MultipleNCO_ExampleScript.py does the same basic method as above but loops through different objective functions for the NCO portion of the testing. This provides multiple options with differing hosting capacity results.

Inputs:

- CYME .sxst study file

Outputs:

- SwitchingDevicesStates_Initial.csv - A CSV file containing the initial states of all switches, reclosers, and breakers in the study, as well as device ID's and device types.
- HCReport_Initial.xlsx - Excel file containing the initial hosting capacity results. This is the same information as in the 'Hosting Capacity Summary Report' obtained through the CYME GUI

- OptReport_Objective_Method.xlsx - Excel file containing the results from the Network Configuration Optimization tool (NCO). This is the same information as in the 'Network Configuration Optimization – Summary' report obtained through the CYME GUI. There will be one of these files for each objective and method run using the Network Configuration Optimization Method. For example, for the Minimize Losses Objective run with the Heuristic Local Method the filename will be 'OptReport_MinimizeLosses_HeuristicLocal.xlsx'. Cases where the NCO tool cannot find a more optimum configuration do not result in an output file.

- SwitchingDevices_AfterOpt_Objective_Method.csv - CSV file containing the states of all switches with the changes applied from the Network Configuration Optimization tool, as well as device ID's and device types. There will be one file for each Objective run using the Network Configuration Optimization tool

- HCReport_Objective_Method.xlsx - Excel file with the hosting capacity results for the new switching device configuration suggested by the Network Configuration Optimization tool. This is the same information as in the 'Hosting Capacity Summary Report' report obtained through the CYME GUI. There will be one file for each Objective run using the Network Configuration Optimization Tool.

[SetSwitchesRunDrive_Script.py](#)

This script loads in a CSV file with a set of switch, recloser, and breaker states, applies those switch states to the study, and then runs EPRI DRIVE on that configuration, with parameters set up to evaluate solar hosting capacity. This particular script enables the user to load in specific configurations of interest to evaluate (e.g., common configurations deployed during maintenance operations), and comes with a CSV file of switching device states that have already been optimized to mitigate abnormal conditions on the original version of the circuit model. This script can be run with the NetwConfOptimiz.sxst tutorial study file which is included with CYME (File > Open Study... > C:\Program Files\CYME\...\tutorial\How-to\NetwConfOptimiz.sxst).

Inputs:

- CYME .sxst study file
- CSV file with a list of Device ID's, Device States, and Device Types

Outputs:

- HCReport_Initial.xlsx - Excel file containing the initial hosting capacity results. This is the same information as in the 'Hosting Capacity Summary Report' obtained through the CYME GUI

[SetSwitches_Script.py](#)

This script loads in a study file, saves the initial switch states, loads in a CSV file with a set of switch, recloser, and breaker states, applies those switch states to the study, and then saves a new study file. This particular script enables the user to load in specific configurations of interest to evaluate (e.g., common configurations deployed during maintenance operations). This script can be run with the 'NetwConfOptimiz.sxst' file included with CYME (File > Open Study... > C:\Program Files\CYME\...\tutorial\How-to\NetwConfOptimiz.sxst)..

Inputs:

- CYME .sxst study file – This can be the NetwConfOptimiz.sxst' study included with CYME
- CSV file with a list of Device ID's, Device States, and Device Types

Outputs:

- SwitchingDevicesStates_Initial.csv – a CSV file with the initial states for the switching devices in the study file, prior to making the manual changes

Adapting the Scripts

One of the main benefits of the scripts is that they can easily be modified to accommodate new functionalities as needs change. Loops could be added to evaluate multiple pre-defined configurations iteratively, the DRIVE module could be replaced with the CYME ICA module, parameters for loads and distributed generators could be changed to evaluate the impacts of seasonality, and so on. Note that the NCO tool does not currently have an option for directly maximizing hosting capacity through an objective function, but multiple objectives can be included in the same optimization, where each is giving a custom weighting factor. So, another area of exploration could be to iterate through different combinations of objectives to find ones that better correlate with hosting capacity.

It is also worth pointing out that the scripts can be used in tandem with the standalone CYME application to leverage the advantages of both methods. While scripting can simplify many time-consuming and repetitive tasks, it can often be easier to make minor modifications to a circuit model manually through the user interface (UI) of the CYME application, which also provides a straightforward means of visualizing results directly on the circuit map. Therefore, at any point in a script, the current version of the circuit model can be saved out and loaded back in through the CYME application to utilize the capabilities of the UI. Alternatively, the CYME application gives the user the ability to create custom reports for any of the built-in tools. So, for example, through the UI, the user could create a custom Load Flow Analysis report that includes 50 unique variables that are not included in any of the default reports, then access the results of that custom report iteratively through a Python script. Note that the ability to leverage the UI and the Python interface concurrently may be limited by the number of licenses available to the user, but the user can always switch back and forth using a single license.

As a general note, there are some CYME parameters that have slightly different names and default settings when accessed through CymPy as compared to accessing them through the CYME application UI. Therefore, it is recommended to directly set each parameter of a particular analysis (or double-check the default values) before running the code.

Additional CymPy Resources

There are a variety of resources available that provide additional documentation and guidance for CymPy. Note: you will need an Eaton/CYME account to access these my.cyme.com links.

The CYME documentation for importing the CymPy package:

https://my.cyme.com/cyme.library/document/show?document%5B_identity%5D=d39ff90c-4e21-6c91-5fba-90958f9760a9

The main CYME and CymPy documentation are here: [https://my.cyme.com/library/f50d3b01-e92d-4127-9940-29bcbad091a9\(v=cyme-9.4\)](https://my.cyme.com/library/f50d3b01-e92d-4127-9940-29bcbad091a9(v=cyme-9.4))

The official distribution of CYME includes three example scripts that use CymPy, which provide additional syntax examples for interacting with other built-in CYME modules. Those scripts can be run directly in CYME using the embedded Python tools or through the standalone environment.

Although the most recent version of CYME is 9.4 (as of this document), interestingly the documentation does not necessarily match between CYME versions on the my.cyme.com site. For example, some aspects of the CymPy documentation are more extensive in the CYME 7.2 documentation. Luckily, the search bar at the top does search through all available CYME versions.

[https://my.cyme.com/library/2a3fba93-9eee-7f3f-e65e-5642b05fc212\(v=cyme-7.2\)](https://my.cyme.com/library/2a3fba93-9eee-7f3f-e65e-5642b05fc212(v=cyme-7.2))

There is a forum available with a sub-forum for Python scripting with CYME:

<https://my.cyme.com/forums/python-scripting>

Currently, however, there is no search bar for the Python scripting sub-forum. One useful work-around is described at the bottom of the post linked below that involves modifying the number of posts shown per page so that the ctrl+F function can be used.

<https://my.cyme.com/forums/python-scripting/20230706110135/searching-through-previous-forum-topics>

The 'auto-complete' functionality in a Python IDE is very helpful in figuring out the class structure and arguments for the CymPy functions and objects.

By default, running EPRI DRIVE also produces a number of intermediate files which are accessible and have more detailed results. Generally, there will be a CYME folder with an EPRI DRIVE subfolder in the Documents folder of the user (e.g., "C:\Users\username\Documents\CYME 9.3\Default 9.0 9.2 9.3\EpriDrive"). Those intermediate files contain the information that is passed back and forth from CYME to the EPRI DRIVE tool and therefore those files can be very useful for debugging purposes.

If a quick search in the documentation does not provide the answers that you need, we strongly recommend going directly to CYME support (cymesupport@eaton.com). Or call +1 (800) 361-3627.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Interconnection Innovation e-Xchange (i2X) program Award Number 41265. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.