

# Hướng dẫn sử dụng Switch 2.0<sup>1</sup>

Trường Chính sách công và Quản lý Fulbright,

Trường Đại học Fulbright Việt Nam

2024

<sup>1</sup>Tổng hợp và biên dịch từ [https://ee.hawaii.edu/public/gfx/contents/mfripp/talks/Fripp\\_2019-08-22\\_EDF\\_Switch\\_Tutorial.pdf](https://ee.hawaii.edu/public/gfx/contents/mfripp/talks/Fripp_2019-08-22_EDF_Switch_Tutorial.pdf)

# Mục lục

<b>1</b>	<b>Hướng dẫn cài đặt các phần mềm liên quan</b>	<b>2</b>
1.1	Anaconda và Python . . . . .	3
1.2	Visual Studio Code . . . . .	3
1.3	Switch, Pyomo, và glpk . . . . .	4
1.4	Tải dữ liệu thực hành . . . . .	5
1.5	Solver nâng cao (không bắt buộc) . . . . .	5
<b>2</b>	<b>Giới thiệu Switch, Pyomo, và Python</b>	<b>7</b>
2.1	Switch . . . . .	7
2.2	Python . . . . .	7
2.3	Optimization và Pyomo . . . . .	7
<b>3</b>	<b>Xây dựng mô hình cơ bản với Switch</b>	<b>9</b>
3.1	Một số tài liệu liên quan . . . . .	9
3.2	Thiết lập module cho mô hình với modules.txt . . . . .	9
3.3	Dữ liệu đầu vào của mô hình . . . . .	10
3.3.1	Mở file .csv trong VS Code . . . . .	10
3.3.2	Các file đầu vào của mô hình . . . . .	12
3.4	Chạy mô hình . . . . .	20
3.5	Xem kết quả . . . . .	21
<b>4</b>	<b>So sánh các kịch bản với Switch</b>	<b>24</b>
4.1	Xác định và chạy các kịch bản . . . . .	24
4.2	Phân tích kết quả . . . . .	28
4.2.1	So sánh chi phí giữa các kịch bản . . . . .	28
4.2.2	Kiểm tra công suất phát theo nguồn trong mỗi kịch bản . . . . .	28
4.2.3	So sánh công suất các loại hình phát điện trong các kịch bản . . . . .	28
<b>5</b>	<b>Tuỳ chỉnh mô hình Switch</b>	<b>31</b>

# Phần 1

## Hướng dẫn cài đặt các phần mềm liên quan

Lưu ý: Trong hướng dẫn này, các bước bắt buộc sẽ được đánh dấu bằng vạch xanh bên trái đoạn văn bản. Các đoạn không có vạch xanh là phần giải thích hoặc không bắt buộc.

Switch phụ thuộc vào các phần mềm mã nguồn mở sau:

- **Visual Studio Code (VS Code)** là ứng dụng soạn thảo code mã nguồn mở. Hướng dẫn này sử dụng VS Code là trình soạn thảo code chính, nhưng bạn hoàn toàn có thể sử dụng một phần mềm soạn thảo code khác.
- **Anaconda** cung cấp phương thức dễ dàng và chuẩn hoá để cài đặt Switch và các phần mềm liên quan khác.
- **Switch** là chương trình giải quyết mô hình tối ưu hóa hệ thống điện bằng cách sử dụng các đầu vào do người dùng cung cấp. Switch là tập hợp các module được viết bằng ngôn ngữ Python, mỗi module mô tả một khía cạnh khác nhau của hệ thống điện.
- **Pyomo** thiết lập khung mô hình tối ưu hoá chung cho Python. Switch sử dụng Pyomo để xác định các thành phần của mô hình cũng như tùy chọn công cụ tối ưu (solver). Pyomo biến mô hình Switch thành định dạng máy tính có thể hiểu, sau đó chuyển mô hình đến **External solver** (vd: glpk, cbc, cplex, hoặc gurobi). External solver sẽ tính toán để tìm ra phương án tối ưu.

Dung lượng ổ cứng: Switch không tốn nhiều dung lượng, nhưng những phần mềm liên quan như Anaconda sẽ cần khoảng 0.3–0.5GB, một số package của Python sẽ tốn khoảng

0.5–1.5 GB. Nếu bạn sử dụng VS Code, bạn sẽ cần thêm 0.3 GB nữa. Lưu ý rằng đây là dung lượng sử dụng cho Mac, nếu bạn sử dụng Windows, dung lượng ổ đĩa cần thiết sẽ cao hơn.

## 1.1 Anaconda và Python

Tải và cài đặt phiên bản “mini” của Anaconda tại <https://docs.conda.io/en/latest/miniconda.html>. Bạn nên chọn phiên bản 64-bit. Trên máy Mac, phiên bản “.pkg” dễ sử dụng hơn “bash”.

Bạn chỉ cần Anaconda phiên bản "mini" để chạy Switch. Nếu bạn muốn sử dụng bản đầy đủ, thay vì sử dụng link trên, bạn có thể tải xuống tại <https://www.anaconda.com/distribution/>. Bản Anaconda đầy đủ sẽ tốn thêm 2–3 GB dung lượng ổ đĩa.

## 1.2 Visual Studio Code

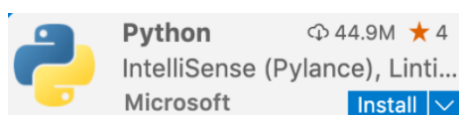
Truy cập <https://code.visualstudio.com/> để tải và cài đặt VS Code. Bạn có thể xem thông tin và hướng dẫn cài đặt VS Code tại <https://code.visualstudio.com/docs/setup/setup-overview>.

Tiếp theo, mở VS Code và làm theo các bước sau để cài đặt extension Python cho VS Code:

- Nhấp vào biểu tượng Extensions (bốn hình vuông) trên Activity Bar ở phía bên trái của cửa sổ Visual Studio Code (hoặc chọn View > Extensions từ menu):



- Nhập “Python” vào thanh tìm kiếm, sau đó chọn “Install” đối với extension mà Microsoft là nhà phát triển:

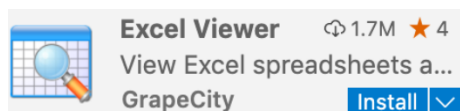


Lặp lại thao tác trên để cài đặt thêm hai tiện ích mở rộng dưới đây:

- **rainbow csv** giúp đọc và chỉnh sửa dữ liệu được lưu trữ trong file csv:



- **excel viewer** cung cấp chế độ xem dạng lưới giống với Excel đối với các file CSV:



**Lưu ý:** Thay đổi Shell trong VS Code:

- **Nếu bạn dùng Windows**, chọn File > Preferences > Settings, sau đó vào Features > Terminal và kéo xuống cho đến khi bạn thấy “Integrated > Default Profile: Windows”, sau đó đổi thành “Command Prompt” thay vì “null” hoặc “PowerShell” (Trong một số cấu hình Windows, PowerShell không chạy đúng bên trong VS Code hoặc không tìm thấy conda environment. Ngoài ra, có trường hợp xuất hiện lệnh “switch” tích hợp có thể làm ẩn lệnh “switch” được sử dụng để chạy Switch. Vì vậy, bạn nên sử dụng “Command Prompt”).
- **Nếu bạn dùng macOS**, chọn Code > Preferences > Settings, sau đó tìm kiếm “terminal heir” và bỏ chọn “Terminal > Integrated: Inherit Env”.

Ngoài ra, bạn nên truy cập File > Preferences > Settings (Windows) hoặc Code > Preferences > Settings (Mac), tìm “open folder” và chọn “on” trong “Window: Open Folders in New Window.”

## 1.3 Switch, Pyomo, và glpk

Khởi động VS Code. Sau đó, chọn View > Command Palette... trên thanh Menu, tìm “Python: Select Interpreter”. Chọn lựa chọn có đề cập đến “(‘base’)” và có “mini-conda3” trong tên đường dẫn. Sau đó chọn Terminal > New Terminal từ thanh menu. Một trang terminal mới dùng để nhập code sẽ hiện ra. Bạn sẽ thấy dòng “conda activate base” (do VS Code nhập), theo sau là dòng lệnh với “(base)” trong dấu nháy. Chạy câu lệnh dưới đây trong terminal và làm theo hướng dẫn hiện trên màn để cài đặt Switch và các phần mềm liên quan:

```
conda create --name switch -c conda-forge switch_model git
```

Để kiểm tra xem tất cả các phần mềm đã được tải xuống hay chưa, hãy chạy các câu lệnh dưới đây trong Command Prompt. Nếu kết quả trả về tên phiên bản của các phần mềm và không báo lỗi, bạn đã cài đặt thành công.

```
conda activate switch
switch --version
pyomo --version
glpsol --version
git --version
```

## 1.4 Tải dữ liệu thực hành

Bạn nên tạo một folder mới để chứa dữ liệu thực hành (ví dụ như Switch Tutorial). Sau đó, trong VS Code, vào File > Open Folder... và chọn thư mục bạn vừa tạo. VS Code sẽ tạo một cửa sổ mới. Nếu bạn nhận được thông báo hỏi bạn có tin tưởng tác giả của các file trong thư mục này không, hãy chọn Có. Tiếp theo, mở một terminal (Terminal > New Terminal), và nhập câu lệnh sau vào terminal để sao chép dữ liệu hướng dẫn vào thư mục hiện tại:

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

Nếu không thể tải xuống dữ liệu thực hành bằng dòng code phía trên, bạn có thể nhập URL vào website, nhấp vào nút "Code" và chọn "Download ZIP". Giải nén tệp .zip và di chuyển thư mục vừa giải nén vào thư mục thực hành. Nếu bạn vào View > Explorer trên thanh menu VS Code, bạn sẽ thấy panel file explorer bên trái cửa sổ VS Code có hai thư mục bạn vừa tải.

## 1.5 Solver nâng cao (không bắt buộc)

Switch sử dụng solver mã nguồn mở glpk. Glpk có thể hoạt động tốt với các mô hình nhỏ, nhưng sẽ không hiệu quả với những mô hình lớn. Đối với bài thực hành trong hướng dẫn này, bạn chỉ cần glpk, nhưng nếu bạn có nhu cầu làm việc với các mô hình lớn hơn, bạn có thể cân nhắc một số solver sau đây:

- COIN CBC (mã nguồn mở, phù hợp nếu không có biến quyết định số nguyên):

```
conda install -c conda-forge coincbc
```

- [CPLEX](#)
- [Gurobi](#) (miễn phí cho giảng viên, sinh viên)

Sau khi tải về, bạn có thể kiểm tra các solver đã được cài trên máy đúng cách hay chưa bằng cách gõ dòng lệnh sau vào command prompt hoặc terminal window:

- CBC

```
cbc  
quit
```

- CPLEX

```
cplex  
quit
```

- Gurobi trên Windows

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

- Gurobi trên Mac hoặc Linux

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

Sau khi cài đặt solver, bạn có thể chọn solver khi chạy Switch bằng cách nhập `--solver cbc`, `--solver cplex` hoặc `--solver gurobi` hoặc trong `options.txt` (sẽ được mô tả trong phần sau).

## Phần 2

# Giới thiệu Switch, Pyomo, và Python

### 2.1 Switch

Bài báo [Switch 2.0: A modern platform for planning high-renewable power systems](#) giới thiệu tổng qua về Switch. Để tìm hiểu nhanh về mô hình, hãy đọc Mục 2. Để tìm hiểu về case study sử dụng trong hướng dẫn này, hãy đọc Mục 3. Nếu bạn muốn tìm hiểu chi tiết mô hình, hãy truy cập vào <https://ars.els-cdn.com/content/image/1-s2.0-S2352711018301547-mmc1.pdf>.

### 2.2 Python

Bạn có thể truy cập hướng dẫn sử dụng Python tại <https://docs.python.org/3/tutorial/>. Để thực hành bài tập trong hướng dẫn này, bạn nên đọc Mục 3 và 4. Nếu bạn muốn tìm hiểu chi tiết hơn, bạn có thể đọc Mục 5 và 6.

### 2.3 Optimization và Pyomo

Pyomo là phần mềm tối ưu hoá mà Switch sử dụng. Một cách đơn giản, Switch áp dụng Pyomo vào các bài tối ưu trong hệ thống điện. Bạn có thể đọc giới thiệu phần mềm tại <https://pyomo.readthedocs.io/>. Trong đó, bạn nên tập trung vào:

- [Tổng quan về Pyomo](#)
- [Các thành phần trong mô hình Pyomo](#)

Trong phần giới thiệu Pyomo, bạn sẽ thấy các phương trình sau:

$$\min c_1x_1 + c_2x_2$$



s.t.

$$a_{11}x_1 + a_{12}x_2 \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 \geq b_2$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Đây là cách thể hiện bài toán tối ưu theo toán học, mang hàm ý “tìm giá trị của  $x_1$  và  $x_2$  sao cho  $c_1x_1 + c_2x_2$  đạt giá trị nhỏ nhất, và  $x_1$  và  $x_2$  thoả mãn các điều kiện ràng buộc”. Trong bài toán này, các giá trị  $x$  được gọi là **biến quyết định** (là con số mà các phần mềm tối ưu sẽ tính toán, ví dụ như lượng điện sản xuất từ một nhà máy trong một giờ), các giá trị  $a, b$  và  $c$  là các **tham số** (dữ liệu đầu vào để xây dựng mô hình, ví dụ như chi phí bảo trì mỗi MWh của một nhà máy),  $c_1x_1 + c_2x_2$  là **hàm mục tiêu** (giá trị lớn nhất hoặc nhỏ nhất), và các (bất) phương trình là các **ràng buộc** (ví dụ như công suất phát không được vượt quá công suất lắp đặt).

## Phần 3

# Xây dựng mô hình cơ bản với Switch

### 3.1 Một số tài liệu liên quan

Một số tài liệu liên quan đến xây dựng mô hình Switch:

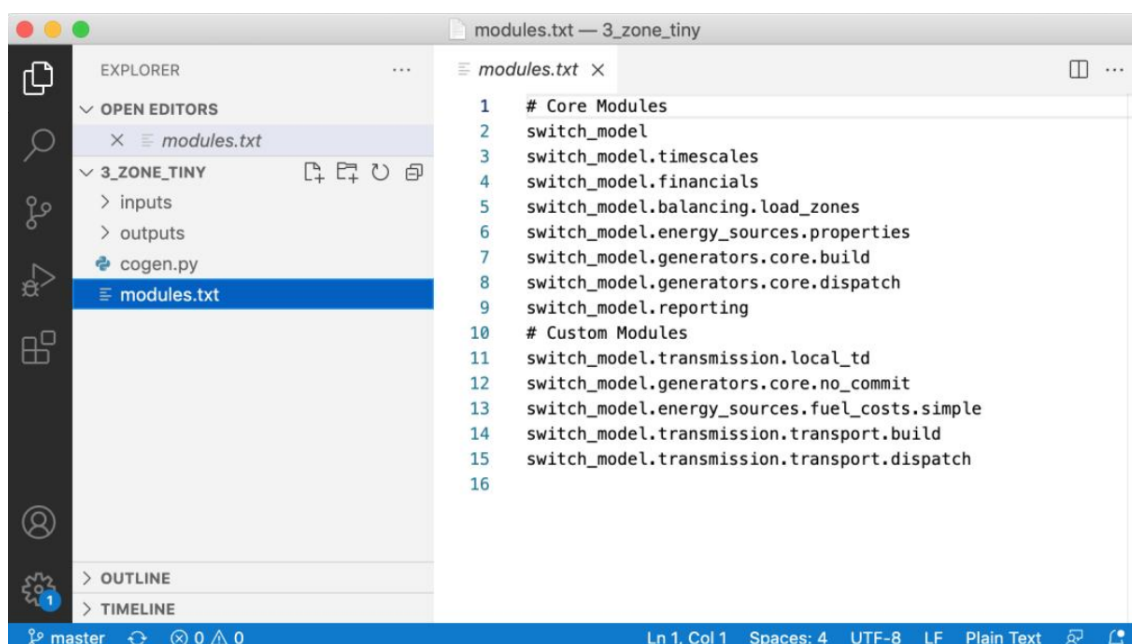
- Website dự án Switch <https://switch-model.org/>, bao gồm tất cả các tài liệu liên quan.
- Bài báo về Switch 2.0 tại <https://doi.org/10.1016/j.softx.2019.100251>, cung cấp cái nhìn tổng quan về Switch 2.0.
- Mục 4 trong <https://ars.els-cdn.com/content/image/1-s2.0-S2352711018301547-mmcl.pdf> giới thiệu mô hình toán của Switch, bao gồm lý thuyết đằng sau các module và các biến quyết định, dữ liệu đầu vào.
- Chạy dòng code “switch --help”, “switch solve --help” (trong thư mục mô hình) hoặc “switch solvescenarios --help” để tìm hiểu về các tùy chọn câu lệnh có sẵn.
- Các bài tập thực hành mô hình nằm trong thư mục “Examples” tại <https://github.com/switch-model/switch>, hoặc <https://github.com/switch-hawaii/>.

### 3.2 Thiết lập module cho mô hình với modules.txt

Chúng ta sẽ làm việc với mô hình đơn giản có tên là “3\_zone\_tiny” bên trong thư mục “switch\_tutorial” mà bạn đã tạo ở phần trước.

Mở VS Code, chọn File > Open Folder... và chọn thư mục 3\_zone\_tiny (chọn thư mục, không phải các file bên trong). Nhấp vào View > Explorer nếu bạn không thấy ngăn Explorer. Nhấp vào “3\_zone\_tiny” trong ngăn Explorer để mở rộng danh sách

thư mục, nhấp vào “inputs”, sau đó vào “modules.txt” bên trong thư mục “inputs”. Thao tác này sẽ tạo ra một cửa sổ tương tự như cửa sổ bên dưới:



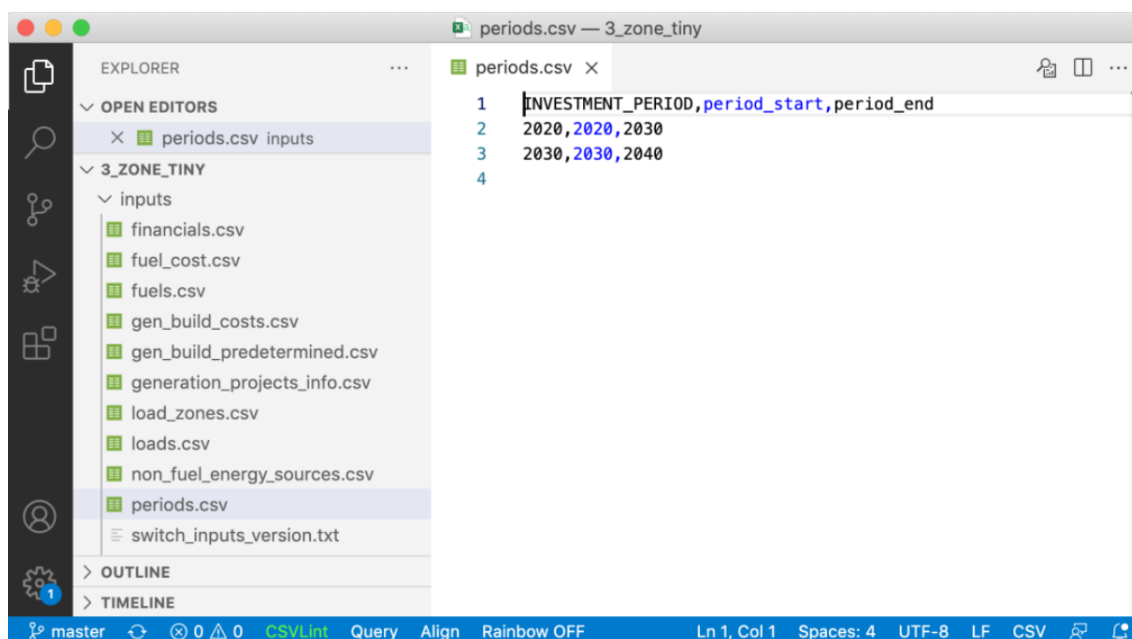
**modules.txt** liệt kê tất cả các module Python sẽ được sử dụng cho bài toán tối ưu “3\_zone\_tiny”. Trong đó, phần **# Core Modules** bao gồm các module bắt buộc, còn **# Custom Modules** liệt kê các module tùy chọn và nâng cao.

## 3.3 Dữ liệu đầu vào của mô hình

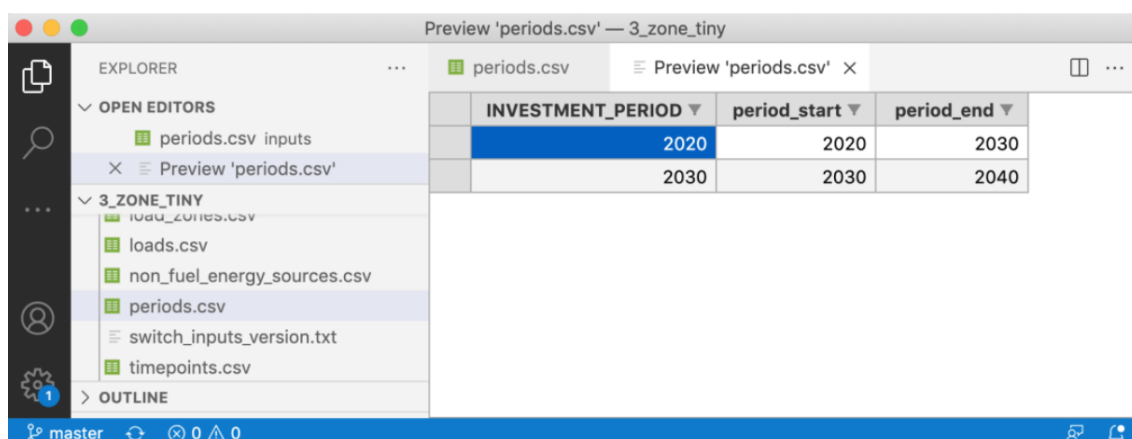
### 3.3.1 Mở file .csv trong VS Code

Trong hướng dẫn này, chúng ta sẽ mở và chỉnh sửa các file .csv trong VS Code bằng extension Rainbow CSV và Excel Viewer. mặc dù vậy, bạn hoàn toàn có thể mở các file này trực tiếp bằng Excel hoặc LibreOffice.

Nhấp vào thư mục “inputs”, chọn file “periods.csv”, bạn sẽ thấy một cửa sổ tương tự như hình bên dưới:



Nhấp vào biểu tượng kính lúp nằm trên hình vuông ở góc trên bên phải chọn (Open Preview), để mở tệp ở chế độ lưới. Bạn cũng có thể nhấp chuột phải vào tab hoặc file periods.csv để chọn “Open Preview”. Nhấp đúp vào đường kẻ dọc giữa các cột để xem đầy đủ giá trị trong mỗi ô. Bạn sẽ thấy một cửa sổ như thế này:



Chế độ xem trước dạng lưới cung cấp các công cụ lọc và sắp xếp đơn giản, ký hiệu bằng các phễu màu xám ở cuối tên mỗi cột. Ở chế độ xem văn bản thông thường, bạn có thể sử dụng một số tính năng ví dụ như ngôn ngữ truy vấn giống SQL có thể truy cập qua View > Command Palette... > Rainbow CSV: RBQL. Bạn có thể tìm tài liệu cho từng tiện ích mở rộng bằng cách sử dụng View > Extensions, sau đó nhấp vào tiện ích mà bạn quan tâm.

### 3.3.2 Các file đầu vào của mô hình

Tất cả các file đầu vào đều có cấu trúc giống nhau. Cột ngoài cùng bên trái là index của dòng, các cột còn lại chứa các giá trị tham số. Thông thường, tên cột index được viết bằng chữ in hoa và tên cột tham số được viết bằng chữ thường.

Một số file đầu vào bắt buộc trong Switch:

- **periods.csv** thể hiện thời gian đầu tư dự kiến. Các dự án hoặc tài sản cố định có thể được xây dựng, mở rộng hoặc hủy bỏ vào đầu mỗi kỳ. Cột đầu tiên (INVESTMENT\_PERIOD) định nghĩa các giai đoạn trong Switch. Period\_start và period\_end thể hiện thời điểm bắt đầu và kết thúc của mỗi kỳ đầu tư. (period\_end có thể là năm cuối cùng của kỳ đó hoặc năm đầu tiên của kỳ tiếp theo. Switch chấp nhận một trong hai và tự dự đoán cách bạn thiết lập tham số này.)

INVESTMENT_PERIOD	period_start	period_end
2020	2017	2026
2030	2027	2036

- **timeseries.csv** liệt kê tất cả các chuỗi thời gian trong mô hình. Mỗi dòng thể hiện một chuỗi thời gian, là tập hợp các mốc thời gian nằm trong một khoảng thời gian đầu tư cụ thể. Ví dụ, trong nhiều mô hình, một chuỗi thời gian sẽ đại diện cho một ngày tương trưng và sẽ gồm 24 mốc thời gian (khoảng cách giữa 2 mốc thời gian liên tiếp là 1 giờ) hoặc 12 mốc thời gian (khoảng cách giữa 2 mốc thời gian liên tiếp là 2 giờ). Cột TIMESERIES là tên chuỗi thời gian, t\_period thể hiện chuỗi thời gian nằm trong kế hoạch đầu tư nào. Cột ts\_duration\_of\_tp thể hiện số mốc thời gian trong một ngày, thường là 24 nếu khoảng thời gian giữa hai mốc liên tiếp là 1 giờ, và 12 nếu khoảng thời gian giữa hai mốc liên tiếp là 2 giờ. Cột ts\_scale\_to\_period thể hiện trọng số của chuỗi thời gian trong toàn bộ thời gian đầu tư (số lần chuỗi thời gian xuất hiện trong thời gian đầu tư). Nếu bạn làm phép tính  $ts\_duration\_of\_tp * ts\_num\_tps * ts\_scale\_to\_period$  cho mỗi chuỗi thời gian, sau đó cộng tất cả các chuỗi thời gian trong mỗi giai đoạn thì tổng này sẽ bằng thời gian của giai đoạn đầu tư đó tính theo giờ (Switch sẽ báo lỗi nếu không bằng). Cách thiết lập chuỗi thời gian cho phép bạn tăng trọng số cho những ngày thông thường và giảm trọng số với những ngày ít phổ biến. Trong hầu hết các trường hợp, giá trị  $ts\_duration\_of\_tp$  và  $ts\_num\_tps$  có giá trị giống nhau trong tất cả chuỗi thời gian. Trong bài thực

hành này, vì số lượng chuỗi thời gian không nhiều, chúng tôi thiết lập các giá trị `ts_duration_of_tp` và `ts_num_tps` khác nhau cho mỗi chuỗi thời gian.

TIMESERIES	ts_period	ts_duration_of_tp	ts_num_tps	ts_scale_to_period
2020_01winter	2020	12	4	913.12
2020_06summer	2020	12	2	1826.25
2030_all	2030	24	1	3652.5

- **timepoints.csv** liệt kê tất cả các mốc thời gian được sử dụng trong Switch. Cột `timepoint_id` thể hiện tên của mốc thời gian (có thể là số hoặc chữ). Cột `timestamp` (không bắt buộc) là nhãn của mốc thời gian. Cột `timeseries` xác định chuỗi thời gian của mốc thời gian.

timepoint_id	timestamp	timeseries
1	2025011500	2020_01winter
2	2025011512	2020_01winter
3	2025011600	2020_01winter
4	2025011612	2020_01winter
5	2025061500	2020_06summer
6	2025061512	2020_06summer
7	2035011512	2030_all

- **load\_zones.csv** thể hiện cấu trúc mạng lưới của hệ thống điện. Trong ví dụ thực hành, chúng tôi sử dụng mô hình ball-and-spoke, mô hình phổ biến nhất trong Switch. Theo đó, điện được truyền tải và phân phối tự do trong mỗi vùng phụ tải, nhưng có thể xảy ra tình trạng tắc nghẽn giữa các vùng phụ tải. Kể cả trong mô hình copperplate, thông tin về vùng phụ tải cũng cần được liệt kê. Cột `LOAD_ZONE` thể hiện tên các vùng phụ tải. Cột `dbid` (không bắt buộc) là nhãn của vùng. Cột

existing\_local\_td và local\_td\_annual\_cost\_per\_mw không bắt buộc, nếu bạn sử dụng module switch\_model.transmission.local\_td, hai tham số này sẽ được dùng để ước tính chi phí nâng cấp đường truyền tải và phân phối cục bộ trong mỗi vùng, dựa trên phụ tải đỉnh cần đáp ứng mỗi năm. Cột local\_td\_loss\_rate thể hiện tỷ lệ tổn thất của quá trình phân phối và truyền tải nội vùng.

LOAD_ZONE	dbid	existing_local_td	local_td_annual_cost_per_mw	local_td_loss_rate
North	1	5.5	66406.5	0.053
Central	2	3.5	61663.4	0.053
South	3	9.5	128040.0	0.053

- **loads.csv** thể hiện nhu cầu phụ tải (tính bằng MW) của mỗi vùng tại mỗi mốc thời gian. Cột LOAD\_ZONE và TIMEPOINT là các cột index, còn cột zone\_demand\_mw thể hiện giá trị phụ tải.

LOAD_ZONE	TIMEPOINT	zone_demand_mw
North	1	5.0
North	2	4.0
North	3	4.5
North	4	4.2
North	5	4.0
North	6	6.0
North	7	6.0

- **gen\_info.csv** liệt kê tất cả các dự án phát điện có thể có trong mô hình, bao gồm các cột:

### Phần 3. Xây dựng mô hình cơ bản với Switch

GENERATION_PROJECT	gen_tech	gen_load_zone	gen_connect_cost_per_mw	gen_capacity_limit_mw	gen_full_load_heat_rate	gen_variable_om	gen_max_age	gen_min_build_capacity	gen_scheduled_outage_rate	gen_forced_outage_rate	gen_is_variable	gen_is_baseload	gen_is_cogen	gen_energy_source	gen_unit_size	gen_ccx_capture_efficiency
N-Geothermal	Geothermal	North	162081.1	1.5	-	28.83	30	0	0.0075	0.0241	0	1	0	Geothermal	-	-
N-Coal_UGCC	Coal_UGCC	North	57566.6	-	7.95	6.0822	40	0	0.08	0.12	0	1	0	Coal	10.0	-
N-Coal_UGCC_CC3	Coal_UGCC_CC3	North	57566.6	-	10.38	9.858	40	0	0.08	0.12	0	1	0	Coal	-	0.85
N-Coal_ST	Coal_ST	North	57566.6	-	9.0	3.4	40	0	0.06	0.1	0	1	0	Coal	-	-
N-NG_GCC	NG_GCC	North	57566.6	-	6.705	3.4151	20	0	0.04	0.06	0	0	0	NaturalGas	-	-
N-NG_GCC_CC3	NG_GCC_CC3	North	57566.6	-	10.08	9.3	20	0	0.04	0.06	0	0	0	NaturalGas	-	0.85
N-NG_ST	NG_ST	North	57566.6	-	10.39	27.807	20	0	0.04	0.06	0	0	0	NaturalGas	-	-
N-Nuclear	Nuclear	North	57566.6	-	9.72	0.0	40	1000	0.04	0.06	0	1	0	Uranium	-	-
N-Biomass_UGCC	Biomass_UGCC	North	57566.6	-	12.5	13.95	40	0	0.09	0.076	0	1	0	BioSolid	-	-
N-Biomass_UGCC_CC3	Biomass_UGCC_CC3	North	57566.6	-	16.3208	20.1307	40	0	0.09	0.076	0	1	0	BioSolid	-	0.85
N-Residential_PV	Residential_PV	North	0.0	1.5	-	0.0	20	0	0.0	0.02	1	0	0	Solar	-	-

Tham số	Định nghĩa
GENERATION_PROJECT	Tên dự án phát điện
gen_tech	Loại hình công nghệ chính của dự án (được sử dụng để báo cáo và/hoặc dùng trong các module tùy chỉnh của người dùng)
gen_load_zone	Vùng phụ tải nơi mà dự án được đặt
gen_connect_cost_per_mw	Chi phí kết nối dự án với mạng lưới truyền tải
gen_capacity_limit_mw	(không bắt buộc) Công suất tối đa có thể được xây dựng, phản ánh giới hạn trong nguồn lực. Nếu không có thông tin, nghĩa là nguồn lực không bị giới hạn
gen_full_load_heat_rate	Lượng năng lượng cần thiết để sản xuất một đơn vị điện, được tính bằng bằng triệu Btu trên MWh. Các dự án NLTT sẽ không có giá trị này
gen_variable_om	Chi phí O&M cho mỗi MWh được sản xuất
gen_max_age	Thời gian vận hành tối đa của dự án; Chi phí sẽ được khấu hao trong thời gian này. Khi dự án đạt đến thời gian vận hành tối đa, nó sẽ tự động ngừng (Switch có thể quyết định xây dựng dự án tương tự để thay thế)
gen_min_build_capacity	(không bắt buộc) Công suất tối thiểu của dự án, có thể là 0 MW hoặc $\geq \text{gen\_min\_build\_capacity}$
gen_scheduled_outage_rate	(không bắt buộc) Tỷ lệ ngừng sản xuất điện để bảo trì theo kế hoạch, được dùng để giảm phụ tải nền (thường chỉ phù hợp với các hệ thống điện lớn)
gen_forced_outage_rate	(không bắt buộc) Tỷ lệ ngừng bắt buộc; dùng để giảm công suất khả dụng của tất cả dự án điện (thường chỉ phù hợp với các hệ thống điện



	lớn)
gen_is_variable	Có phải dự án năng lượng biến đổi hay không (ví dụ như gió hoặc mặt trời). Công suất khả dụng của dự án được mô tả trong file <code>variable_capacity_factors.csv</code> . Nhận giá trị 0 hoặc 1
gen_is_baseload	Có phải dự án chạy nền (sản lượng phát không đổi) hay không. Nhận giá trị 0 hoặc 1
gen_is_cogen	Không cần đối với các module bắt buộc, nhưng có thể cần trong các module tùy chỉnh
gen_energy_source	Tên nguồn năng lượng sử dụng trong dự án (hoá thạch hoặc tái tạo). Nếu dự án sử dụng nhiều nguồn khác nhau, danh sách nguồn năng lượng cần được cập nhật đầy đủ trong file <code>gen_multiple_fuels.csv</code>
gen_unit_size	(không bắt buộc) Công suất cố định, bổ sung cho công suất sẵn hoặc cho điều động tổ máy. Không có hiệu lực trừ khi module <code>generators.core.gen_build_discrete</code> và/hoặc <code>generators.core.commit_discrete</code> được sử dụng
gen_ccs_capture_efficiency	(không bắt buộc) Tỷ lệ khí thải $CO_2$ được CCS giữ lại
gen_ccs_energy_load	(không bắt buộc) Tỷ lệ sản lượng điện mất đi do vận hành CCS
gen_storage_efficiency	(không bắt buộc) Hiệu suất chu chuyển (round trip efficiency) của hệ thống lưu trữ
gen_store_to_release_ratio	(không bắt buộc) Công suất lưu trữ thực của hệ thống lưu trữ, được đo bằng tỷ lệ khi so với công suất phát
gen_storage_energy_to_power_ratio	(không bắt buộc) Tỷ lệ cố định của sản lượng lưu trữ (MWh) trên một đơn vị công suất lắp đặt (MW), tức thời gian lưu trữ. Switch sẽ tối ưu tham số này nếu không được đề cập
gen_storage_max_cycles_per_year	(không bắt buộc) Số lượng chu kỳ sạc/xả đầy tối đa mỗi năm cho dự án lưu trữ
gen_min_uptime	(không bắt buộc) Số giờ tối thiểu một máy phát

	điện phải hoạt động sau khi khởi động
gen_min_downtime	(không bắt buộc) Số giờ tối thiểu một máy phát điện phải duy trì trạng thái nghỉ sau khi tắt
gen_startup_fuel	(không bắt buộc) Lượng nhiên liệu cần thiết để khởi động máy phát điện (triệu Btu/MW công suất cần khởi động)

- **gen\_build\_costs.csv** thể hiện chi phí đầu tư thêm một MWh vào dự án tương ứng. Cột gen\_overnight\_cost là chi phí qua đêm, còn cột gen\_fixed\_om là chi phí cố định O&M. Nó cũng hiển thị chi phí mở rộng công suất các dự án lưu trữ nếu có (gen\_storage\_energy\_overnight\_cost, \$/MWh). Chỉ có những dự án có trong file này mới có thể được đầu tư mở rộng công suất.

GENERATION_PROJECT	build_year	gen_overnight_cost	gen_fixed_om
N-Coal_ST	1995	2687700.0	21390.0
N-Geothermal	2000	5524200.0	0.0
N-NG_CC	2008	1143900.0	5868.3
N-NG_GT	2009	605430.0	4891.8

- **gen\_build\_predetermined.csv** liệt kê tất cả dự án đã được mở rộng công suất ở thời điểm trước khi nghiên cứu bắt đầu. Cột build\_year thể hiện năm công suất tăng thêm được đầu tư, còn cột build\_gen\_predetermined thể hiện mức công suất được đầu tư (tính bằng MW). Khi các công suất tăng thêm đạt đến tuổi giới hạn, nó sẽ tự động ngừng hoạt động.

GENERATION_PROJECT	build_year	build_gen_predetermined
N-Coal_ST	1995	2
N-Geothermal	2000	1
N-NG_CC	2008	2

- **gen\_multiple\_fuels.csv** chỉ cần thiết nếu một dự án phát điện sử dụng nhiều loại nhiên liệu khác nhau. Tập này có hai cột, generation\_project và fuel. Chỉ có các dự án có giá trị gen\_energy\_source = “multiple” trong file gen\_info.csv mới được liệt kê trong file này.
- **fuel\_cost.csv** xác định nguồn nhiên liệu khả dụng tại mỗi vùng phụ tải trong từng giai đoạn nghiên cứu. Cột fuel\_cost thể hiện chi phí nhiên liệu, tính bằng USD/triệu Btu. File này được sử dụng trong module switch\_model.energy\_sources.fuel\_costs.simple. Ngoài ra, bạn cũng có thể sử dụng module switch\_model.energy\_sources.fuel\_costs.markets, lúc này bạn cần xác định đường cung nhiên liệu bằng cách nhập thông tin vào file fuel\_supply\_curves.csv.

load_zone	fuel	period	fuel_cost
North	Uranium	2020	2.19
Central	Uranium	2020	2.19
South	Uranium	2020	2.19

- **fuels.csv** liệt kê tất cả các nguồn nhiên liệu hoá thạch có trong mô hình của bạn. Cột co2\_intensity và upstream\_co2\_intensity không bắt buộc, trong đó cột đầu

tiên thể hiện mức phát thải  $CO_2$  tính theo tấn trên một triệu Btu đốt cháy, cột thứ hai thể hiện cường độ  $CO_2$  upstream. Ngoài ra, bạn cũng có thể thêm cột thể hiện nhiên liệu có thoả điều kiện RPS hay không (giá trị 0 hoặc 1).

fuel	co2_intensity	upstream_co2_intensity
Coal	0.09552	0.0
ResidualFuelOil	0.0788	0.0
DistillateFuelOil	0.07315	0.0
NaturalGas	0.05306	0.0

- **non\_fuel\_energy\_sources.csv** liệt kê tất cả các nguồn năng lượng không có trong fuels.csv, ví dụ: năng lượng tái tạo. Cột gen\_energy\_source trong file gen\_info.csv phải được liệt kê trong file fuels.csv hoặc non\_fuel\_energy\_sources.csv.

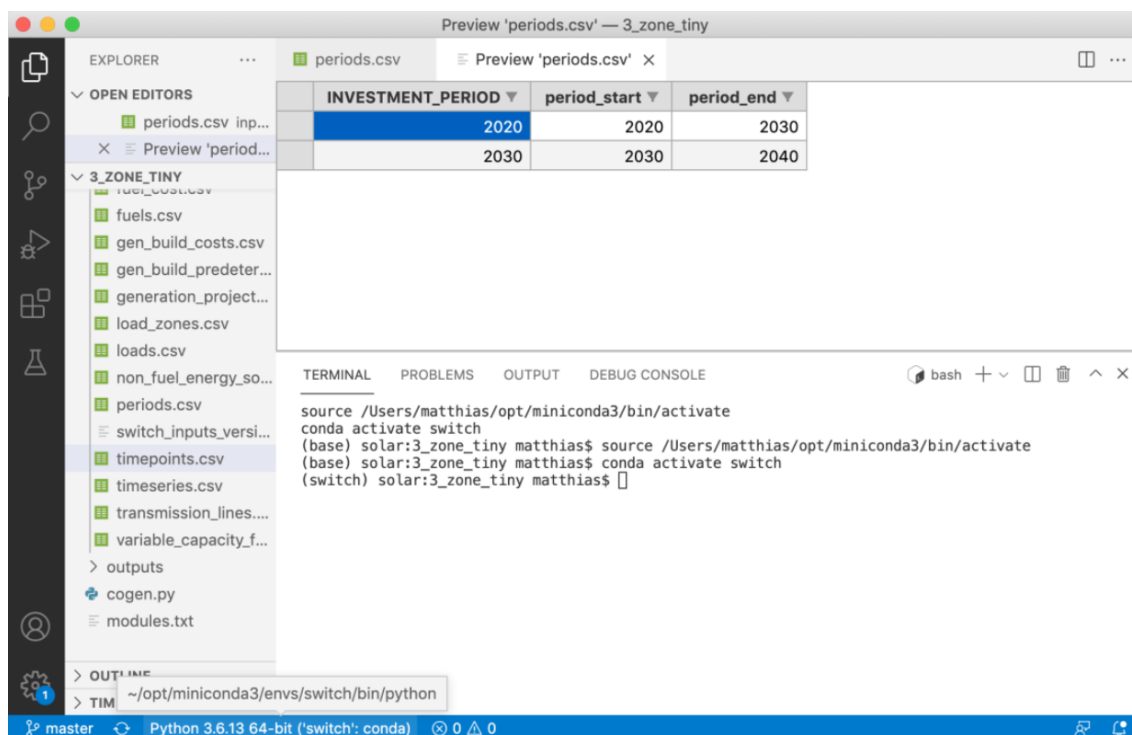
```
energy_source
Wind
Solar
Geothermal
Water
Electricity
```

- **financials.csv** định nghĩa tất cả các tham số tài chính của mô hình. Tất cả các chi phí được chiết khấu đến năm tài chính cơ sở bằng cách sử dụng giá trị của cột discount\_rate. Chi phí vốn được khấu hao trong suốt vòng đời của dự án bằng cách sử dụng interest\_rate (nếu không có interest\_rate, interest\_rate sẽ mặc định bằng discount\_rate). Các tham số này được dùng cho tất cả các dự án.

base_financial_year	discount_rate	interest_rate
2015	0.05	0.07

## 3.4 Chạy mô hình

Trong VS Code, chọn View > Command Palette... > Python: Select Interpreter. Sau đó chọn interpreter ('switch': conda). Interpreter này phải nằm trong /opt/miniconda3/envs/switch/bin/python. Bạn chỉ cần thực hiện thao tác này khi mở thư mục trong VS Code lần đầu tiên. Tiếp theo, chọn Terminal > New Terminal. Thao tác này sẽ mở một trang terminal ở cuối cửa sổ VS Code, nơi bạn có thể thêm các câu lệnh. Cửa sổ của bạn sẽ trông giống như thế này:



Trong quá trình cài đặt các phần mềm liên quan ở phần 1, nếu bạn không thể chạy conda trong VS Code, bạn có thể mở Terminal.app (Mac) hoặc Anaconda Command Prompt (Windows), chạy “conda activate switch”, sau đó sử dụng lệnh “cd” để điều hướng đến thư mục 3\_zone\_tiny, và chạy tất cả câu lệnh trong hướng dẫn này như bình thường.

Gõ câu lệnh này vào terminal:

```
switch --help
```

Bạn sẽ thấy danh sách các tùy chọn câu lệnh (ví dụ: “switch solve”). Để đọc thêm thông tin về câu lệnh này, bạn cần nhập:

```
switch solve --help
```

Bạn sẽ thấy tất cả các tùy chọn bạn có thể với Switch hoặc với bất kỳ module nào. Một số câu lệnh cho phép bạn thêm hoặc xóa các module có trong file modules.txt, thay đổi lượng thông tin hiển thị trong lúc chạy, xác định solver, thay đổi thư mục lưu đầu vào và đầu ra, lưu các file tạm thời,...Chúng ta sẽ thử sử dụng tùy chọn “--verbose” (hiện nhiều thông báo hơn), “--stream-solver” (hiện thông báo từ solver) và “--sorted-output” (dữ liệu trong file đầu ra được sắp xếp theo thứ tự):

```
switch solve --verbose --stream-solver --sorted-output
```

Nếu mô hình chạy bình thường, kết quả đầu ra sẽ được lưu vào thư mục hiện đã được ghi vào thư mục “outputs”.

## 3.5 Xem kết quả

Nhấp vào thư mục “outputs” trong ngăn Explorer bên trái. Bạn sẽ thấy khoảng 20 file kết quả, chủ yếu là các tệp .csv. Chữ cái hoa mỗi từ trong tên file sẽ được viết hoa. Cũng giống như file đầu vào, cột đầu tiên của file kết quả sẽ là cột index, sau đó đến các cột giá trị. Các file kết quả này được module switch\_model.reporting tạo ra.

Một số file kết quả chính bao gồm:

- **BuildGen.csv** gồm cột index (GEN\_BLD\_YRS) và một cột hiển thị các giá trị do Switch tính toán. Một số dự án mở rộng công suất bao gồm xây dựng thêm 11 MW C-Coal\_IGCC (than IGCC trong vùng phụ tải miền Trung - Central) vào năm 2020, hay 4 MW C-Wind-1 trong năm 2030. Lưu ý rằng file này chỉ tính toán công suất tăng thêm, không tính đến công suất bị sa thải hoặc công suất tích lũy.
- **BuildTx.csv** thể hiện công suất cần bổ sung vào công suất truyền tải hiện tại, ví dụ như cần bổ sung thêm 5.8 MW công suất truyền tải Trung tâm - miền Nam (Central - South) và 5.2 MW công suất truyền tải miền Bắc - miền Trung (North - Central) vào năm 2020.

Một số file kết quả khác:

- **BuildLocalTD** hiển thị lượng công suất T&D nội vùng cần bổ sung ở mỗi vùng trong từng giai đoạn (thường là đủ để đáp ứng phụ tải đỉnh của nguồn điện phân tán).
- **BuildMinGen** là biến nhị phân, được sử dụng để xác định một dự án nên giữ nguyên công suất hiện tại hay đầu tư mở rộng công suất.
- **DispatchBaseloadByPeriod** hiển thị mức sản lượng phát của các dự án phụ tải nền, không đổi trong toàn bộ một khoảng thời gian.
- **DispatchTx** hiển thị mức độ truyền tải trên mỗi hệ thống kết nối liên vùng tại từng thời điểm.
- **GenFuelUseRate** hiển thị lượng nhiên liệu (tính bằng MMBtu) mỗi dự án sử dụng trong từng thời điểm.
- **WithdrawFromCentralGrid** hiển thị phụ tải trong mỗi vùng tại mỗi thời điểm, sau khi trừ đi các tài nguyên được phân phối.

Trong câu lệnh solver, bạn có thể thêm “--save-expressions all” để lưu các giá trị trung gian được đặt tên mà Switch tính toán và sử dụng. Đó là tất cả các Biểu thức (Expression) Pyomo dựa trên các biến quyết định và dữ liệu đầu vào của bạn.

Các kết quả đầu ra của Switch rất chi tiết. Nếu bạn cần thông tin tổng hợp, bạn có thể sử dụng các phần mềm phân tích dữ liệu khác (Excel, R, Python,...). Switch cũng cung cấp một số kết quả tổng hợp được bởi `switch_model.reporting`, ví dụ như:

- **total\_cost.txt** thể hiện giá trị hiện tại ròng (NPV) của dự án trong toàn bộ thời gian nghiên cứu (USD 126,750,492). Lưu ý rằng mặc dù đầu vào mô hình chỉ bao gồm một số ít ngày tượng trưng, các giá trị này được thêm trọng số để có thể tính toán được chi phí hoạt động cho toàn bộ thời gian nghiên cứu.
- **cost\_components.csv** phân tổng chi phí thành chi phí cố định cho máy phát điện, chi phí truyền tải và phân phối nội vùng và liên vùng, cả chi phí biến đổi O&M và nhiên liệu.
- **gen\_cap.csv** hiển thị công suất lắp đặt tích lũy, bao gồm cả công suất xây dựng thêm và công suất tới hạn cho mỗi dự án trong từng giai đoạn đầu tư. File `gen_cap.csv` cũng hiển thị một số đặc điểm khác của dự án, ví dụ như công nghệ phát điện, vùng phụ tải, vốn và chi phí O&M.

Một số kết quả tổng hợp khác bao gồm:

- **costs\_itemized.csv** hiển thị thành phần chi phí giống như **cost\_components.csv** nhưng được phân chia theo giai đoạn nghiên cứu.
- **dispatch\_annual\_summary.csv** hiển thị tổng sản lượng sản xuất, chi phí biến đổi và lượng phát thải, được nhóm theo thời kỳ, công nghệ và nguồn năng lượng chính. Trong khi đó, **dispatch\_zonal\_annual\_summary.csv** hiển thị cùng dữ liệu nhưng phân chia theo vùng phụ tải.
- **dispatch.csv** hiển thị sản lượng sản xuất, chi phí biến đổi và lượng phát thải, phân theo dự án, nhiên liệu, và thời điểm, có bao gồm trọng số để phản ánh tỷ trọng của thời điểm trong một năm thông thường.
- **dispatch\_wide.csv** hiển thị sản lượng điện của từng dự án phát điện tại mỗi thời điểm, với cột thể hiện dự án, còn dòng thể hiện thời điểm.
- **electricity\_cost.csv** hiển thị tổng chi phí NPV và chi phí thực tế trong mỗi kỳ, cũng như mức tiêu thụ điện và chi phí cho mỗi MWh truyền tải.
- **load\_balance.csv** hiển thị tất cả các lần điện vào và ra tại nút truyền tải của mỗi vùng trong mỗi thời điểm, bao gồm tổng sản lượng điện, lượng vào ròng, phụ tải (trừ năng lượng cho phân phối) và bất kỳ nguồn hoặc bồn chứa nào do người dùng chỉ định.



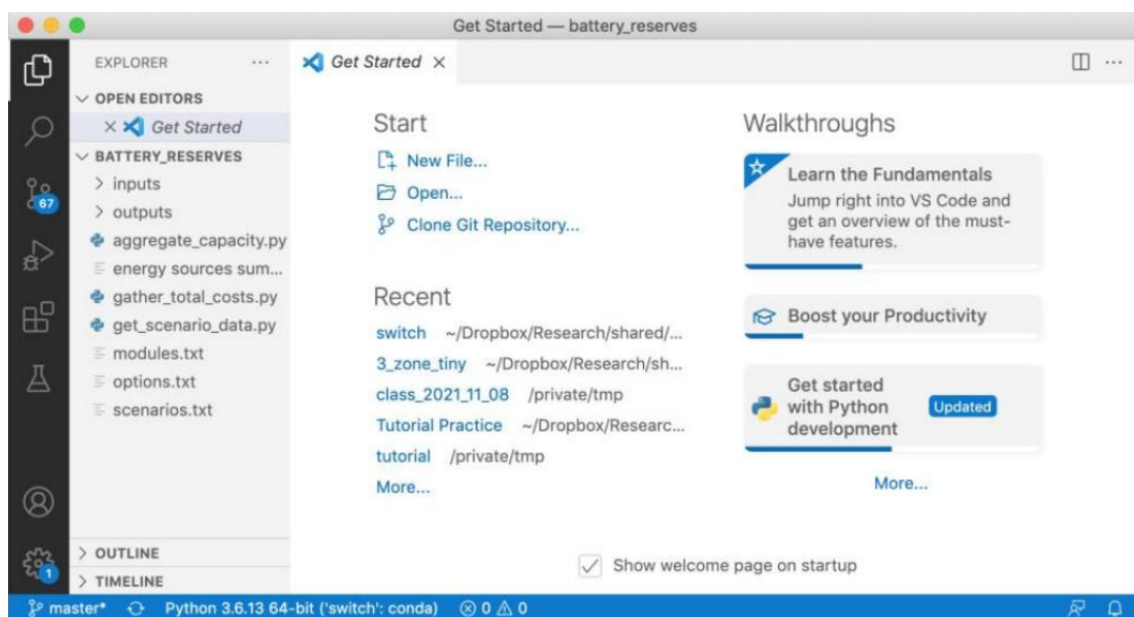
## Phần 4

# So sánh các kịch bản với Switch

Trong phần này, để minh họa rõ nét hơn các kịch bản khác nhau trong Switch, chúng ta sẽ dùng một ví dụ phức tạp hơn ví dụ 3\_zone\_tiny sử dụng ở các phần trước.

### 4.1 Xác định và chạy các kịch bản

Trong VS Code, sử dụng File > Open Folder... để mở thư mục battery\_reserves bên trong thư mục switch\_tutorial mà bạn đã tạo trong phần 1. Chọn View > Command Palette... > Python: Select Interpreter > “Python <version> ('switch')” trong VS Code để thiết lập interpreter cho thư mục này. Bạn sẽ thấy một cửa sổ tương tự như bên dưới:



Mở `modules.txt` trong thư mục `inputs > regulation`. Mô hình này sử dụng các module tùy chọn nâng cao hơn so với mô hình thực hành trước. Ví dụ, mô hình sử dụng module `energy_sources.fuel_costs.markets` nhằm định nghĩa đường cung nhiên liệu thay vì chi phí cố định đơn giản như trong `energy_sources.fuel_costs.simple`. Các module `generators.core.commit.operate` và `generators.core.commit.fuel_use` mô tả chi tiết hoạt động điều độ tổ máy, bao gồm tình trạng thiếu hiệu quả của `part-load`, thay vì chỉ điều độ như trong `generators.core.no_commit`.

Mô hình này cũng sử dụng một số module tùy chọn khác như:

- **`generators.core.gen_discrete_build`** và **`generators.core.commit.discrete`** buộc các dự án được xây dựng và cam kết thành từng phần riêng biệt tương ứng với một đơn vị phát điện hoàn chỉnh (một dự án duy nhất có thể đại diện cho nhiều đơn vị phát điện giống hệt nhau).
- **`generators.extensions.storage`** cho phép các dự án phát điện là dự án lưu trữ, dòng điện xoay chiều, có khả năng lưu trữ và có tổn thất.
- **`balancing.operating_reserves_areas`** và **`balancing.operating_reserves.spinning_reserves_advanced`** xác định nguồn và yêu cầu của dự phòng quay.
- Tập hợp các module từ subpackage **`hawaii`**, cung cấp các hành vi thực nghiệm mới (mở rộng thị trường nhiên liệu thông qua một terminal LNG) hoặc các quy tắc và hành vi cụ thể của đảo Hawaii (USA).

Module `spinning_reserves_advanced` định nghĩa hai loại dự phòng: dự phòng sự cố được sử dụng để bù đắp nếu một nhà máy điện lớn ngừng hoạt động và dự phòng điều tiết được sử dụng để cân bằng hệ thống theo chu kỳ thời gian dưới thị trường. Nhìn chung, dự phòng sự cố dễ cung cấp hơn dự phòng điều tiết vì chúng ít được sử dụng hơn và ít yêu cầu giao tiếp hơn (chỉ cần theo dõi tần suất hệ thống). Khi sử dụng module này, người dùng cần xác định loại dự phòng mà mỗi dự án có thể cung cấp (theo mặc định, tất cả các dự án đều có thể cung cấp dự phòng quay), và xác định sản phẩm nào có thể được dùng để cung cấp dự phòng sự cố hoặc điều tiết. Người dùng cũng cần xác định các nguyên tắc cho từng loại dự phòng.

Trong `options.txt`, bạn sẽ thấy một số thiết lập như:

```
--contingency-reserve-type contingency
--regulating-reserve-type regulation
```

```
--spinning-requirement-rule Hawaii
--unit-contingency
```

Các thiết lập phía trên mang ý nghĩa rằng đoạn lệnh phía trước được cung cấp thông qua sản phẩm có tên là chữ cuối cùng của đoạn code. Ví dụ, dự phòng sự cố được cung cấp thông qua sản phẩm “contingency”. Bạn có thể tìm hiểu các thiết lập bằng cách chạy “switch solve –help” trong thư mục battery\_reserves, xem qua code của spinning\_reserves\_advanced, hoặc đọc Supplementary Material từ bài báo Switch 2.0.

Trong thư mục “inputs”, bạn sẽ tìm thấy ba thư mục chứa ba bộ dữ liệu đầu vào khác nhau. Các bộ dữ liệu chỉ khác nhau ở các quy tắc về việc dự án nào có dự phòng quay, đã được liệt kê trong file generation\_projects\_reserve\_capability.csv. Đầu tiên, hãy xem qua thư mục **inputs/none**. Tất cả các nhà máy quy mô tiện ích, bao gồm cả điện gió và điện mặt trời, đều có thể cung cấp dự phòng điều tiết và dự phòng sự cố, tùy thuộc vào công suất phát điện khả dụng. Ngoài ra còn có pin dự phòng sự cố chuyên dụng (Oahu\_Battery\_Conting) chỉ có thể cung cấp dự phòng sự cố và một pin dự phòng điều tiết (Oahu\_Battery\_Reg) có thể cung cấp cả dự phòng sự cố và dự phòng điều tiết. Tuy nhiên, pin lưu trữ quy mô lớn (Oahu\_Battery\_4 và Oahu\_Battery\_6) không cung cấp bất kỳ dịch vụ dự trữ nào. Bạn có thể xem thêm chi tiết về từng loại pin trong file gen\_info.csv. Pin dự phòng được mô hình hóa như nguồn điện thuần túy, không có chức năng lưu trữ, trong khi pin lưu trữ quy mô lớn được thiết kế với dung lượng lưu trữ cố định là 4 hoặc 6 giờ.) Trong **inputs/contingency**, các thông số kỹ thuật giống nhau trong tương tự với trường hợp trên, ngoại trừ pin Oahu\_Battery\_4 và Oahu\_Battery\_6 có thể cung cấp dự phòng sự cố nhưng không cung cấp dự phòng điều tiết. Cuối cùng, trong **inputs/contingency**, pin lưu trữ quy mô lớn có thể cung cấp cả dự phòng sự cố và điều tiết.

Để so sánh các kịch bản với nhau, bạn có thể thiết lập các tùy chọn trong file options.txt, sau đó trong terminal, bạn chỉ cần chạy “switch solve”. Mặc dù vậy, cách làm này đòi hỏi bạn phải điều chỉnh các thiết lập thủ công, cũng như bạn khó kiểm soát được những kịch bản đã chạy. Vì vậy, chúng tôi khuyên bạn nên thiết lập các kịch bản với “switch solve” trong terminal.

Để chạy các kịch bản trong ví dụ này, hãy chạy file scenarios.txt trong VS Code.

Chi tiết 5 kịch bản:

- **battery\_bulk**: Pin cung cấp lưu trữ nhưng không cung cấp dự phòng, điều chỉnh phụ tải không bao gồm chuyển dịch phụ tải và dự phòng, thiết lập tỷ lệ phụ tải có thể hoãn ở mức 0%, và bắt buộc EV sạc theo lịch trình thông thường. Các flag cần

thiết cho kịch bản bao gồm: `--inputs-dir inputs/none --demand-response-reserve-types none --ev-reserve-types none`. Ngoài ra, chúng tôi cũng đặt `--outputs-dir outputs/battery_bulk` để tách kết quả của kịch bản này với những kịch bản khác.

- **battery\_bulk\_and\_conting**: Tương tự kịch bản trên, ngoại trừ việc pin lưu trữ có thể cung cấp dự phòng sự cố. Các thiết lập flag nhìn chung cũng giống như kịch bản trên, nhưng file đầu vào sẽ được lấy từ `--inputs-dir inputs/contingency`.
- **battery\_bulk\_and\_reg**: Giống như kịch bản trên, pin dự trữ ngoài cung cấp dự phòng sự cố còn có thể cung cấp dự phòng điều tiết. Các thiết lập flag tương tự, nhưng sử dụng đầu vào trong `--inputs-dir inputs/regulation`.
- **dr\_bulk**: Tương tự kịch bản `battery_bulk_and_reg`, nhưng 10% nhu cầu phụ tải và tất cả các EV đều có thể cung cấp dịch chuyển phụ tải, nhưng không cung cấp dự phòng. Các thiết lập flag nhìn chung giống với `battery_bulk_and_reg` với điều chỉnh nhỏ ở `--demand-response-share 0.1 --ev-timing optimal`.
- **dr\_bulk\_and\_reserves**: Đây là kịch bản tối ưu (và thực tế giống như mô hình mặc định), trong đó pin và phản hồi phụ tải đều có thể cung cấp dịch chuyển phụ tải trọng lớn, dự phòng sự cố, và dự phòng điều tiết. Flag tương tự `dr_bulk` nhưng với `--demand-response-reserve-types regulation contingency --ev-reserve-types regulation contingency`.

Mô hình này hiện đang được giải quyết bằng gurobi. Nếu bạn chưa cài Gurobi, hãy mở `options.txt` trong VS Code, đặt dấu “#” trước “`-solver gurobi`” để phần mềm bỏ qua dòng này khi chạy. Nếu bạn đã cài đặt COIN CBC, bạn có thể thêm “`-solvercbc`” vào `options.txt`. Nếu không, Switch sẽ tự động sử dụng glpk.

Mở một cửa sổ terminal và sử dụng “`cd`” để điều hướng đến thư mục “`battery_reserves`”. Khi bạn đã ở trong thư mục `battery_reserves`, hãy chạy:

```
switch solve-scenarios
```

Lệnh này yêu cầu Switch giải quyết lần lượt tất cả các tình huống bạn đã xác định. Ở đây chúng ta sẽ so sánh tổng chi phí cho mỗi khách hàng trong cả 5 tình huống để xem các biện pháp can thiệp khác nhau có giá trị như thế nào.

Bạn có thể giải quyết nhiều tình huống cùng lúc trên siêu máy tính—`switch solve-scenarios` có thể chạy trên các máy riêng biệt, miễn là chúng chia sẻ cùng một thư mục. Nếu bạn cài đặt `openmpi` (“`conda install openmpi`”), bạn cũng có thể khởi chạy nhiều

phiên bản switch bằng lệnh duy nhất: “mpirun switch solve-scenarios”. Lưu ý rằng kết quả từ các kịch bản khác nhau có thể bị trộn vào nhau, do đó, bạn có thể sử dụng tùy chọn --log-run để lưu trữ kết quả cho từng kịch bản trong một file riêng biệt. Bạn cũng có thể sử dụng cài đặt --logs-dir khác nhau cho từng kịch bản.

Các mô hình này tương đối lớn nên cần hơn một giờ để chạy xong nếu như bạn sử dụng solver thương mại như gurobi hoặc cplex. Nếu bạn sử dụng glpk hoặc cbc, thời gian sẽ lâu hơn đáng kể.

## 4.2 Phân tích kết quả

Kết quả từ cả năm kịch bản đều được lưu trong các thư mục con của battery\_reserves/outputs, với tên trùng với tên kịch bản. Các file này là file .csv, do đó bạn có thể phân tích chúng bằng hầu hết các phần mềm phân tích dữ liệu như Excel, R, hay Python. Trong phần thực hành này, chúng tôi sẽ sử dụng Excel.

### 4.2.1 So sánh chi phí giữa các kịch bản

Để thu thập thông tin chi phí, bạn cần vào thư mục “outputs”, chọn từng thư mục ứng với từng kịch bản, và mở file “total\_cost.txt”. File này thể hiện tổng chi phí đã chiết khấu của kịch bản.

### 4.2.2 Kiểm tra công suất phát theo nguồn trong mỗi kịch bản

Mở file “energy sources summary.xlsm” trong thư mục “battery\_reserves/outputs”. Sheet “**data**” sao chép dữ liệu từ file “energy\_sources\_<tên kịch bản>.csv” của tất cả kịch bản và tổng hợp.

Trong sheet “**graphs**”, bạn sẽ thấy biểu đồ công suất phát của hệ thống điện năm 2045. Để thay đổi nội dung hiển thị biểu đồ, bạn cần chọn sheet “**filtered**”, sau đó chọn nội dung bạn muốn tổng hợp.

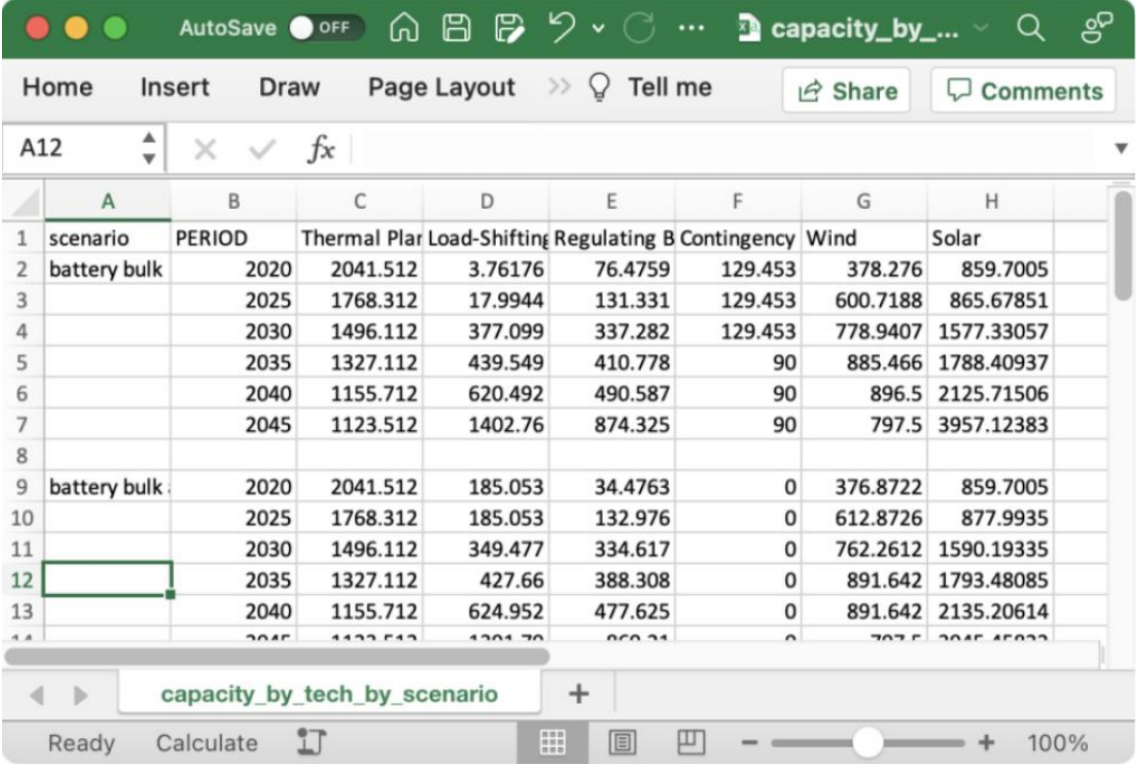
### 4.2.3 So sánh công suất các loại hình phát điện trong các kịch bản

Quay lại cửa sổ terminal của VS Code trong thư mục “reserve\_study”. Để tổng hợp dữ liệu đầu tư công suất, hãy nhập câu lệnh sau:

```
python aggregate_capacity.py
```

Bạn sẽ thấy thông báo rằng capacity\_by\_tech\_by\_scenario.csv đã được lưu trong thư mục đầu ra. Câu lệnh này sử dụng package Python “Pandas” để thu thập dữ liệu từ các tệp gen\_cap.csv trong toàn bộ thư mục kết quả, tổng hợp dữ liệu, sau đó định dạng lại dữ liệu để có thể dễ dàng vẽ biểu đồ trong Excel.

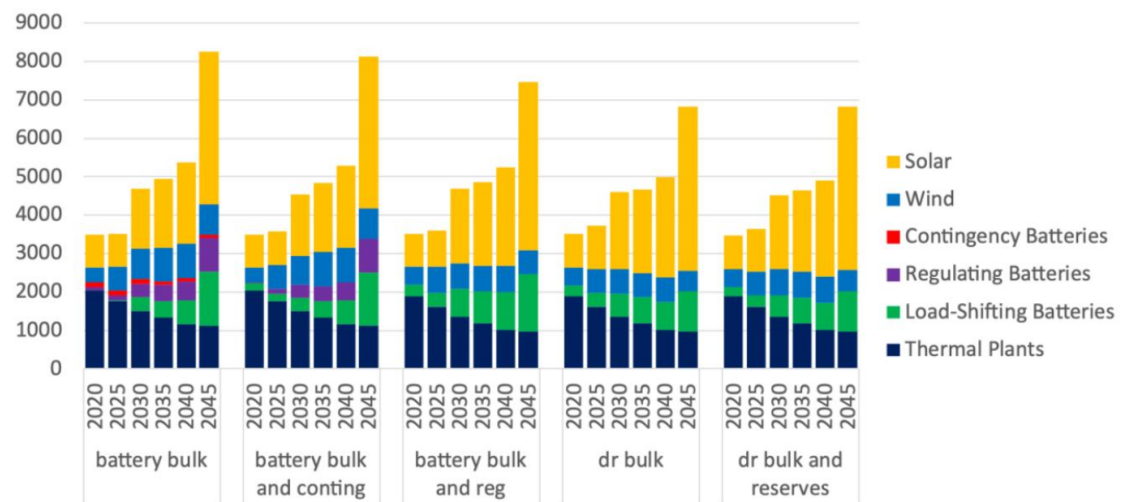
Mở file capacity\_by\_tech\_by\_scenario.csv trong Excel. Bạn sẽ thấy file có nội dung như thế này:



The screenshot shows an Excel spreadsheet titled 'capacity\_by\_tech\_by\_scenario'. The spreadsheet has columns for scenario, period, and various technologies. The data is organized into two main sections: 'battery bulk' and 'battery bulk' (repeated). The first section shows data for periods 2020, 2025, 2030, 2035, 2040, and 2045. The second section shows data for periods 2020, 2025, 2030, 2035, and 2040. The technologies included are Thermal Plan, Load-Shifting, Regulating B, Contingency, Wind, and Solar. The values represent capacity in different units.

	A	B	C	D	E	F	G	H
1	scenario	PERIOD	Thermal Plan	Load-Shifting	Regulating B	Contingency	Wind	Solar
2	battery bulk	2020	2041.512	3.76176	76.4759	129.453	378.276	859.7005
3		2025	1768.312	17.9944	131.331	129.453	600.7188	865.67851
4		2030	1496.112	377.099	337.282	129.453	778.9407	1577.33057
5		2035	1327.112	439.549	410.778	90	885.466	1788.40937
6		2040	1155.712	620.492	490.587	90	896.5	2125.71506
7		2045	1123.512	1402.76	874.325	90	797.5	3957.12383
8								
9	battery bulk	2020	2041.512	185.053	34.4763	0	376.8722	859.7005
10		2025	1768.312	185.053	132.976	0	612.8726	877.9935
11		2030	1496.112	349.477	334.617	0	762.2612	1590.19335
12		2035	1327.112	427.66	388.308	0	891.642	1793.48085
13		2040	1155.712	624.952	477.625	0	891.642	2135.20614
14		2045	1123.512	1402.76	874.325	0	797.5	3957.12383

Bạn có thể tùy ý tổng hợp và vẽ biểu đồ. Một ví dụ về biểu đồ bạn có thể vẽ:



## Phần 5

# Tuỳ chỉnh mô hình Switch

Trong phần này, chúng ta sẽ xem cách viết tùy chỉnh một module trong mô hình 3\_zone\_tiny. Chúng ta sẽ giả định một công nghệ đồng phát sản xuất điện từ nhiệt thải có thể đặt ở một nhà máy nhiệt điện bất kỳ. Công nghệ này sẽ có các đặc điểm sau:

- Nhiệt thải được chuyển đổi thành điện năng giảm tải ở mức 20 MMBtu trên MWh;
- Chi phí tồn kho là USD 150,000 cho mỗi MW công suất phát mỗi năm, bao gồm cả thu hồi vốn và O&M;
- Một khi đã xây dựng thì không thể sa thải công suất (điều này giúp tính toán công suất khả dụng và chi phí dễ dàng hơn);
- Nếu không cần thiết có thể loại bỏ việc giảm tải đồng phát.

Trong ví dụ này, chúng ta cần hai biến quyết định mới: Lượng công suất đồng phát tại mỗi nhà máy nhiệt điện trong mỗi giai đoạn, và lượng điện sản xuất từ thiết bị đồng phát trong mỗi thời điểm. Ràng buộc về việc sản lượng điện không vượt quá công suất lắp đặt hoặc nhiệt thải có sẵn cũng cần được thêm vào mô hình. Sản lượng tăng thêm từ quá trình cũng cần được thêm vào phương trình cân bằng phụ tải, và chi phí mua thiết bị đồng phát cũng cần được thêm vào chi phí hệ thống.

Trong VS Code, vào File > Open Folder... để mở mô hình 3\_zone\_tiny. Chọn File > New. Chúng ta sẽ lưu trữ các tham số cho mô hình tại đây. Ở dòng đầu tiên, nhập tên tham số mới, phân cách bằng dấu phẩy, không có khoảng trắng hoặc dấu ngoặc kép: “cogen\_heat\_rate,cogen\_fixed\_cost”. Ở dòng thứ hai, nhập các giá trị: “20,150000”. Sau đó, chọn File > Save và lưu dưới dạng “cogen.csv” bên trong thư mục “inputs”.



Mở “cogen.py” trong thư mục “3\_zone\_tiny” và xem qua các dòng code trong file.

Câu lệnh chính của module nằm trong hàm bắt đầu bằng:

```
def define_components(m):
```

Câu lệnh tạo ra hàm “define\_components” chấp nhận mô hình có tên là “m”. Switch sẽ gọi hàm này khi xây dựng mô hình và hàm này sẽ gắn các thành phần cogen vào mô hình.

Thông thường, các tham số của mô hình (thành phần dữ liệu) sẽ được định nghĩa:

```
m.cogen_heat_rate = Param()
m.cogen_fixed_cost = Param()
```

Lưu ý rằng trong Switch, các set đều được viết in hoa, các tham số (như cogen\_heat\_rate) được viết bằng chữ thường, các biến và biểu thức được viết bằng chữ thường có chữ cái đầu in hoa, còn các ràng buộc được viết bằng chữ thường có chữ cái đầu in hoa, giữa các từ là dấu “\_”.

Bây giờ chúng ta cần định nghĩa một biến BuildCogen cho mỗi nhà máy nhiệt điện trong mỗi giai đoạn nghiên cứu và một biến DispatchCogen cho mỗi nhà máy nhiệt điện cho mỗi mốc thời gian. Nếu bạn đọc Supplementary Material của bài báo Switch, bạn sẽ thấy rằng Switch chia máy phát điện thành loại sử dụng nhiên liệu và không sử dụng nhiên liệu, trong đó máy phát điện chạy nhiên liệu gọi là FUEL\_BASED\_GENS. Với trường hợp đồng phát, chúng ta sẽ tập trung vào máy phát điện chạy bằng nhiên liệu. Các biến được định nghĩa như sau:

```
m.BuildCogen = Var(m.FUEL_BASED_GENS, m.PERIODS,
                    within=NonNegativeReals)
m.DispatchCogen = Var(m.FUEL_BASED_GENS, m.TIMEPOINTS,
                       within=NonNegativeReals)
```

Bây giờ chúng ta cần tính toán lượng công suất đồng phát sẽ sản xuất tại mỗi dự án trong mỗi giai đoạn:

```
def CogenCapacity_rule(m, g, p):
    capacity = sum(
        m.BuildCogen[g, p2] for p2 in m.PERIODS if p2 <= p
    )
    return capacity
m.CogenCapacity = Expression(
    m.FUEL_BASED_GENS, m.PERIODS,
```

```

        rule=CogenCapacity_rule
    )

```

Trong đó g thể hiện dự án, còn p thể hiện giai đoạn.

Sản lượng phát của đồng phát không được vượt quá khả năng công suất đặt và nhiệt thải:

```

def Max_DispatchCogen_rule(m, g, t):
    test = (m.DispatchCogen[g, t] <= m.CogenCapacity[g, m.tp_period[t]])
    return test
m.Max_DispatchCogen = Constraint(
    m.FUEL_BASED_GENS, m.TIMEPOINTS, rule=Max_DispatchCogen_rule
)

```

Sản lượng phát của đồng phát bị giới hạn bởi tốc độ nhiệt và nhiệt thải. Nếu bạn xem code `switch_model.generators.core.no_commit` hoặc Supporting Material, bạn sẽ thấy nhiên liệu được sử dụng trong module `no_commit` là `DispatchGen * gen_full_load_heat_rate` (MMBtu). Chúng ta cũng có thể chuyển đổi công suất đầu ra thành MMBtu bằng cách nhân `DispatchGen` với 3.412, là lượng MMBtu trong 1 MWh. Chúng ta cũng có thể tính được lượng nhiệt cần thiết cho một tổ máy đồng phát bằng cách lấy `DispatchCogen * cogen_heat_rate`. Các dòng code dưới đây thể hiện rằng mức tiêu thụ nhiệt của tổ máy đồng phát không được lớn hơn nhiệt thải khả dụng:

```

def DispatchCogen_Available_Heat_rule(m, g, t):
    if (g, t) in m.GEN_TPS:
        # this is a timepoint when the thermal plant can run
        heat_input = m.DispatchGen[g, t] * m.gen_full_load_heat_rate[g]
        work_done = m.DispatchGen[g, t] * 3.412 # power output in MMBtu
    else:
        # thermal plant is pre-construction or retired
        heat_input = 0
        work_done = 0
    rule = (
        m.DispatchCogen[g, t] * m.cogen_heat_rate # heat used by cogen
        <= heat_input - work_done # heat available
    )
    return rule
m.DispatchCogen_Available_Heat = Constraint(
    m.FUEL_BASED_GENS, m.TIMEPOINTS,

```

```
rule=DispatchCogen_Available_Heat_rule
)
```

Thêm sản lượng từ đồng phát vào phương trình cân bằng phụ tải:

```
# calculate power output from cogen units in zone z in timepoint t
def CogenZonalOutput_rule(m, z, t):
    total_output = sum(
        m.DispatchCogen[g, t]
        for g in m.FUEL_BASED_GENS
        if m.gen_load_zone[g] == z
    )
    return total_output
m.CogenZonalOutput = Expression(
    m.LOAD_ZONES, m.TIMEPOINTS,
    rule=CogenZonalOutput_rule
)
# Add cogen output to system generation balance
m.Zone_Power_Injections.append('CogenZonalOutput')
```

Cuối cùng, chúng ta cần thêm chi phí của tổ máy cogen vào tổng chi phí hệ thống. Ở đây, chúng tôi chỉ trình bày cấu trúc chi phí đơn giản, chỉ bao gồm chi phí cố định hàng năm dựa trên tổng công suất lắp đặt:

```
# Calculate fixed costs for all cogen units online in period p
def CogenFixedCost_rule(m, p):
    total_capacity = sum(m.CogenCapacity[g, p]
                        for g in m.FUEL_BASED_GENS)
    return total_capacity * m.cogen_fixed_cost
# Add fixed costs to model
m.CogenFixedCost = Expression(m.PERIODS, rule=CogenFixedCost_rule)
m.Cost_Components_Per_Period.append('CogenFixedCost')
```

Chúng ta cũng cần một đoạn code nhỏ để đọc các tham số `cogen_heat_rate` và `cogen_fixed_cost` từ `cogen.csv`:

```
def load_inputs(m, switch_data, inputs_dir):
    switch_data.load_aug(
        filename=os.path.join(inputs_dir, 'cogen.csv'),
        autoselect=True,
```

```
param=(m.cogen_heat_rate, m.cogen_fixed_cost))
```

Để chạy mô hình, đầu tiên, bạn cần mở `modules.txt` trong thư mục `3_zone_tiny` và thêm một dòng mới có nội dung “cogen”, hoặc nhập câu lệnh “`--include-module cogen`”.

Đến trang Terminal hoặc Anaconda Command Prompt, điều hướng đến thư mục `3_zone_tiny`, và chạy:

```
switch solve --outputs-dir outputs_cogen
```

Thao tác này sẽ chạy mô hình của bạn và lưu kết quả vào thư mục `outputs_cogen`.

Bây giờ bạn có thể sử dụng VS Code (hoặc lệnh “`fc`” trên Windows hoặc lệnh “`diff`” trên các nền tảng khác) để so sánh kết quả giữa kịch bản hiện tại và với thư mục “`outputs`” ban đầu. Bạn sẽ thấy rằng đồng phát đã giảm `total_cost` từ USD 127 triệu xuống còn USD 118 triệu. Bạn cũng sẽ thấy rằng công nghệ này chủ yếu được thêm vào các nhà máy C-Coal\_IGCC và C-Coal\_ST. Nó cũng khiến việc xây dựng C-Coal\_IGCC vào năm 2020 giảm từ 11.1 MW xuống 8.2 MW và khiến N-Wind-1 tăng nhẹ (xem `BuildGen.csv`). Nguyên nhân có thể là do đồng phát điều độ được trong khi IGCC cần phải chạy hết công suất trong toàn bộ thời gian.