PREP-SHOT Documentation

Release latest

Zhanwei Liu and Xiaogang He

Nov 20, 2025

CONTENTS

Authors

Zhanwei Liu (liuzhanwei@u.nus.edu), Xiaogang He (hexg@nus.edu.sg)

Contributors

Bo Xu (xubo_water@dlut.edu.cn), Jingkai Xie (jingkai@nus.edu.sg), Shuyue Yan (shuyue.yan@u.nus.edu) , Zhouyan Li (zhouyan@nus.edu.sg), Quan Yuan (quanyuan@nus.edu.sg), Kewei Zhang (kewei_zhang@u.nus.edu), Yaozhong Cui (cuiyaozhong@u.nus.edu)

Organization

National University of Singapore

Version

latest

Date

Nov 20, 2025

CHAPTER

ONE

OVERVIEW

PREP-SHOT (**P**athways for **R**enewable **E**nergy **P**lanning coupling **S**hort-term **H**ydropower **O**pera**T**ion) is a transparent, modular, and open-source energy expansion model, offering advanced solutions for multi-scale, intertemporal, and cost-effective expansion of energy systems and transmission lines.

The model sets itself apart from existing energy expansion models through its deeper consideration of hydropower processes. While models such as urbs might treat hydropower as fixed processes, and others like GenX and PLEXOS may not fully capture the dynamic nature of water heads or consolidate multiple hydropower stations into a single unit, PREP-SHOT is uniquely designed to address these oversights.

Our model explicitly considers the plant-level water head dynamics (i.e., time-varying water head and storage) and the system-level network topology of all hydropower stations within a regional grid. This results in a more accurate reflection of the multi-scale dynamic feedbacks between hydropower operation and energy system expansion. Furthermore, it enables the realistic simulation of the magnitude and spatial-temporal variability of hydropower output, particularly in regions with a large number of cascade hydropower stations.

With PREP-SHOT, we aim to answer key questions related to the future of energy planning and utilization:

- How can we effectively plan an energy portfolio and new transmission capacity under deep uncertainty?
- How can we quantify the impacts of variable hydropower on the generation and capacity of future energy portfolios?

CHAPTER

TWO

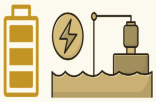
HOW IT WORKS

PREP-SHOT: **P**athways for **R**enewable **E**nergy **P**lanning coupling **S**hort-term **H**ydropower **O**pera**T**ion

I Prepare Inputs

Hydropower

- Cascade topology
- Water travel time
- Initial water head
- Initial & Terminal storage
- Natural inflow
- Storage bounds
- Ramping rate
- Outflow bounds
- Output bounds
- Output efficiency
- Stage-Storage curve
- Tailwater rating curve
- Water head loss coefficient



Storage technology

- Initial & Terminal energy storage level
- Discharging & Charging efficiency
- Power to energy ratio
- Discharging & Charging output bounds

Non-dispatchable technology (Solar & Wind)

- Capacity factor
- Installed upper bound

Dispatchable technology (Coal & Nuclear)

- Ramping rate
- Power output bounds
- Carbon dioxide emission per unit of electricity
- Fuel cost per unit of electricity

Transmission line

- Transmission topology
- Transmission efficiency

Cost-related parameters

- Discount rate
- Unit investment cost
- Unit fixed Operation and Maintenance (O&M) cost
- Unit variable O&M cost
- Lifetime of technologies and transmission lines
- Capacity-Age relationship (only to generation technology)

Non-cost parameters

- Electricity demand
- Planning horizon
- Representative periods
- Time step
- Others (see details in Supplementary Note 1 in Liu and He (2023))

II Build Model

Objective function

- Minimize the cost of the whole energy system

Constraints

- Lifetime constraints
- Carbon emission constraints
- Power balance constraints
- Transmission constraints
- Power output constraints
- Power output variation constraints
- Energy storage constraints
- Water balance constraints
- Reservoir outflow constraints
- Reservoir storage constraints

III Solve Model

Software

- GUROBI

Algorithms

- Simplex method
- Barrier method
- Simulation-based iterative method (see details in Supplementary Figure 14 in Liu and He (2023))

IV Analyze Results

Model output

- Capacity of newly built technology per modelled year per zone
- Capacity of newly built transmission lines per modelled year per zone
- Transmitted power per modelled year between zones
- Generation of each technology and discharging and charging of each storage per modelled year per zone
- Generation flow, withdrawal water flow and spillage flow of each hydropower station per modelled year

Source: Liu and He (2023).

CHAPTER

THREE

KEY FEATURES

- PREP-SHOT is an optimization model based on linear programming for energy systems with multiple zones.
- It aims to minimize costs while meeting the given demand time series.
- By default, it operates on hourly-spaced time steps, but this can be adjusted.
- The input data is in Excel format, while output data is generated in a NetCDF format using Xarray.
- It supports multiple types of solvers such as HiGHS , GUROBI, COPT, and MOSEK via PyOptInterface.
- It allows the input of multiple scenarios for specific parameters.
- As a pure Python program, it benefits from the use of pandas and Xarray, simplifying complex data analysis and promoting extensibility.

CHAPTER

FOUR

OFFLINE DOCUMENTATION

To browse the documentation offline, you can download a PDF copy for offline reading (Synchronize updates with online documentation).

4.1 Installation

This page provides instructions on how to install and use PREP-SHOT. The installation process is divided into the following steps:

4.1.1 Step 1: Download PREP-SHOT

Ensure you have downloaded the PREP-SHOT model from the GitHub repository.

You may either clone the repository using the command:

```
git clone https://github.com/PREP-NexT/PREP-SHOT.git
```

or download the repository as a zip file here.

4.1.2 Step 2: Install dependencies

Assuming you have already installed Python, the `requirements.txt` file contains all the dependencies for the project (default install open-source solver *HiGHS*). I also recommend create a new environment for PREP-SHOT and installing the dependencies within the new environement. This approach isolates the project and its dependencies, helping to prevent conflicts with other Python projects.

```
cd PREP-SHOT
conda create -n prep-shot python=3.8
conda activate prep-shot
pip install -r requirements.txt
```

4.1.3 Step 3: Run an example (Optional)

Once the environment is activated, you can run an example of *Tutorial* with the following command:

```
python run.py
```

You can also run examples using Jupyter notebooks located in the *./example/* directory.

PREP-SHOT default solve models using open-source solver HiGHS. also support commercial solvers, including Gurobi, COPT and MOSEK. They offer academic licenses. To use these solvers, you need to install them and modify the solver in the *config.json* file.

4.1.4 Step 4: Run your own model

You can prepare your input data referring to the example in the *input* and *southeast* folder. The detailed input data are introduced in the *Tutorial*. After preparing the input data, you can modify the *config.json* file to set the solver and other parameters. Then you can run your model with the following command:

```
python run.py
```

4.2 Model Inputs/Outputs

The model requires several input parameters, provided via input files. These parameters, their dimensions, and descriptions are as follows:

4.2.1 Inputs

The parameters used, their desciptions, and their input file name in the model are as follows:

Table 1: Parameters

Parameter [Unit]	Description	Input file[Page 12, 1]
historical capacity[2] [MW]	The capacity of each technology in each zone for each year, taking into account the number of years that each technology has been in operation starting from the beginning of the planning period.	historical_capacity
capacity factor [N/A]	Capacity factor of different non-dispatchable technologies.	capacity_factor
carbon emission limit [tCO2]	Carbon emission limit of different zones.	carbon_emission_limit
emission factor [tCO2/MWh]	Emission factor of different technologies.	emission_factor
water delay time [N/A]	Water delay time of connection between reservoirs.	water_delay_time
demand [MW]	Demand of different balancing authorities.	demand
discount factor [N/A]	Discount factor for each year.	discount_factor
distance [km]	Distance of different pair of zones.	distance
discharge efficiency [N/A]	Discharge efficiency of storage technologies.	discharge_efficiency

continues on next page

Table 1 – continued from previous page

Parameter [Unit]	Description	Input file[Page 12, 1]
charge efficiency [N/A]	Charge efficiency of storage technologies.	charge_ efficiency
energy to power ratio [MWh/MW]	Power to energy ratio ratio of storage technologies.	energy_to_ power_ratio
fuel price [dollar/MWh]	Fuel price of different technologies.	fuel_price
Predefined hydropower[3] [MW]	Predefined hydropower output of all reservoirs.	predefined_hydropower
inflow [m3/s]	Inflow of all reservoirs.	inflow
initial energy storage level [1/MWh]	Initial energy storage level of different storage technologies.	initial_energy_ storage_level
lifetime [yr]	Lifetime of different technologies.	lifetime
new technology lower bound [MW]	Lower bound of newly-built installed capacity of different technologies for each investment year.	new_technology_ lower_bound
new technology upper bound [MW]	Upper bound of newly-built installed capacity of different technologies for each investment year.	new_technology_ upper_bound
ramp down [1/MW]	Ramp down rate of different technologies.	ramp_down
ramp up [1/MW]	Ramp up rate of different technologies.	ramp_up
reservoir characteristics [As per data sheet]	Reservoir characteristics data includes designed water head, maximum storage, minimum storage, operational efficiency, area of affiliation, installed capacity, maximum power output, minimum power output, maximum outflow, minimum outflow, and maximum generation outflow.	reservoir_ characteristics
reservoir storage upper bound [m3]	Upper bound of volume of hydropower reservoirs.	reservoir_storage _upper_bound
reservoir storage lower bound [m3]	Lower bound of volume of hydropower reservoirs.	reservoir_storage_ lower_bound
final reservoir storage level [m3]	Final volume of hydropower reservoirs.	final_reservoir_ storage_level
initial reservoir storage level [m3]	Initial volume of hydropower reservoirs.	initial_reservoir_ storage_level
technology fixed OM cost [dollar/MW/yr]	Fixed operation and maintenance cost of different technologies.	technology_fixed_ OM_cost
technology variable OM cost [dollar/MW/yr]	Variable operation and maintenance costs of different technologies.	technology_variable_ OM_cost
technology investment cost [dollar/MW]	Investment cost of different technologies.	technology_investment _cost
technology portfolio [MW]	Existing total installed capacity across all zones.	technology_ portfolio
technology upper bound[4] [MW]	Upper bound of installed capacity of different technologies.	technology_upper_ bound
transmission line existing capacity [MW]	Capacity of existing transmission lines (if there is no exising nor planned transmission lines between two specific zones, leave the data entries blank).	transmission_line_ existing_capacity

continues on next page

Table 1 – continued from previous page

Parameter [Unit]	Description	Input file[Page 12, 1]
transmission line efficiency [N/A]	Efficiency of transmission lines across all zones.	transmission_line_efficiency
transmission line fixed OM cost [dollar/MW/yr]	Fixed operation and maintenance costs of transmission lines.	transmission_line_fixed_OM_cost
transmission line variable OM cost [dollar/MW/yr]	Variable operations and maintenance costs of transmission lines.	transmission_line_variable_cost
transmission line investment cost [dollar/MW/km]	Investment cost of transmission lines.	transmission_line_investment_cost
transmission line lifetime [yr]	Lifetime of transmission lines.	transmission_line_lifetime
technology type [N/A]	Categories of different technologies.	technology_type
reservoir tailrace level-discharge function [m & m3/s]	Relationship between tailrace level and total discharge for different reservoirs.	reservoir_tailrace_level_discharge_function
reservoir forebay level-volume function [m & m3]	Relationship between forebay level and volume for different reservoirs	reservoir_forebay_level_volume_function

Note:

- *inf* refers to Infinity, indicating that there is no upper bound.
- *None* refers to a null value for current item.

[1] The input files format is `.xlsx`.

[2] For instance, assuming the planning period spans from 2020 to 2050, with 2020 being the starting point, let's consider a technology that has been in operation since 2019. In this case, 2020 would mark its 2nd year of operation within the planning period. These inputs are useful for modelling the retirement of existing technologies.

[3] To model the simplified hydropower operation.

[4] To model the potential of technologies with land, fuel, and water constraints.

4.2.2 Outputs

The output of the model is stored in a NetCDF file, please refer to this simple tutorial and official documentation of Xarray to understand how to manipulate NetCDF files.

The output file contains the following variables:

Table 2: Output Variables

Variable name [Unit]	Description
trans_import [MW]	The electrical power transmitted from Zone 1 and effectively received by Zone 2 through the transmission line, after adjusting for transmission losses.
trans_export [MW]	The electrical power initially sent out by Zone 1 for transmission to Zone 2 via the transmission line, before adjusting for any transmission and distribution losses during its journey to Zone 2.
gen [MW]	Power output of different technologies during user-defined time interval.
install [MW]	Existing installed capacity of different technologies.
carbon [Ton]	Carbon emissions across different years.
charge [MW]	Charged electricity of different storage technologies.
cost [dollar]	Total cost (including total investment cost, total variable OM cost, and total fixed OM cost) over the planning period.
cost_breakdown [dollar]	Breakdown of total cost (including total investment cost, total variable OM cost, and total fixed OM cost) over the planning period, by zone, year, and technology.
cost_var [dollar]	Total variable OM cost (including technology variable OM cost, transmission line variable OM cost, and fuel cost) over the planning period.
cost_var_breakdown [dollar]	Breakdown of total variable OM cost (including technology variable OM cost, transmission line variable OM cost, and fuel cost) over the planning period, by zone, year, and technology.
cost_fix [dollar]	Total fixed OM cost (including technology fixed OM cost, transmission line fixed OM cost) over the planning period.
cost_fix_breakdown [dollar]	Breakdown of total fixed OM cost (including technology fixed OM cost, transmission line fixed OM cost) over the planning period, by zone, year, and technology.
cost_newtech [dollar]	Total investment cost of technologies over the planning period.
cost_newtech_breakdown [dollar]	Breakdown of total investment cost of technologies over the planning period, by zone, year, and technology.
cost_newline [dollar]	Investment cost of transmission lines over the planning period.
cost_newline_breakdown [dollar]	Breakdown of total investment cost of transmission lines over the planning period, by zone, year, and technology.
income [dollar]	Saved cost due to abstracted water resources over the planning period.
genflow [m3/s]	Generated water flow of different reservoirs.
spillflow [m3/s]	Spilled water flow of different reservoirs.

4.2.3 Execute various scenarios

By employing command-line parameters, you can execute different scenarios using the model. For example, if you wish to run a scenario referred to as "low demand," you can prepare input data named `demand_low.xlsx`. Subsequently, when running the model, you can utilize command-line parameters to specify the scenario value. For instance, you can execute the model by executing the command `python run.py --demand=low`.

4.2.4 Setting global parameters

This section will guide you on how to tune the PREP-SHOT model parameters to compute the energy system for your needs. After you have prepared your input data based on the previous sections, you can proceed to tune the model parameters before you run it.

Within the root directory of the model, you will find a JSON file containing the parameters that you can tune for the model, named `config.json`. This file contains the following parameters:

Model Parameter	Description
input_folder	Specifies the name of the folder containing the input data.
output_filename	Specifies the name of the output file.
hour	Specifies the number of hours in each time period.
month	Specifies the number of months in each time period.
dt	Specifies the timestep for the simulation in hours.
hours_in_year	Specifies the number of hours in a year. Typically, this is set to 8760.
isinflow	Specifies whether to include inflow in the optimization problem. It can be used by assigning *isinflow = true* or *false*.
error_threshold	Specifies the error threshold for the model, while iterating for a solution. This parameter controls the convergence of the hydro model.
iteration_number	Specifies the maximum number of iterations for the hydro model, while iterating for a solution.
solver	Specifies the solver to be used for the optimization problem.
solver_path	Specifies the path of the dynamic library of solver. This is required while automatic detection of the installation directory of the solver fails. Please provide refer to PyOptInterface documentation.
solver_parameters	Specifies the solver-specific parameters for mosek, gurobi, highs, and copt.

After you have tuned the parameters, you can run the model by following the steps in the *Installation* page.

You can also try out the model with the sample data provided in the `input` folder. Refer to the *Model Inputs/Outputs* page for a walkthrough of this example, inspried by real-world data.

4.3 Tutorial

In this tutorial, we'll guide you through running your first PREP-SHOT model! This example will illustrate an electricity capacity expansion scenario.

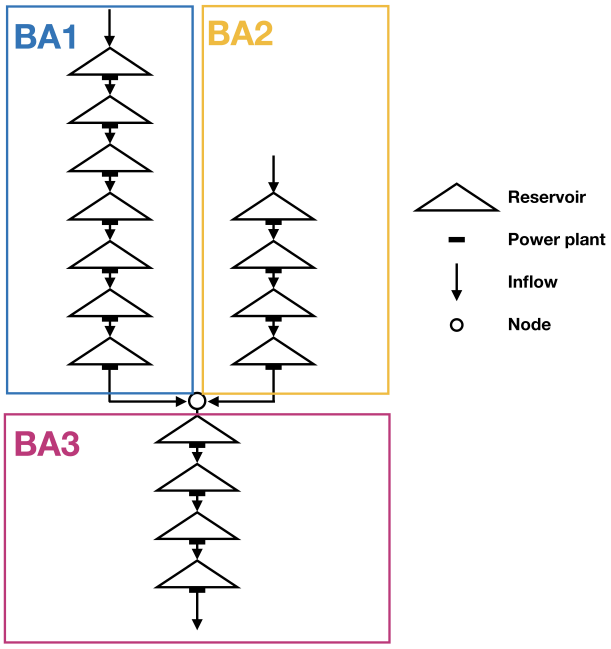
4.3.1 Scenario Background

This scenario is inspired by real-world data, drawing primarily from the following resources:

- U.S. Energy Information Administration (EIA)
- U.S. Army Corps of Engineers (USACE)
- U.S. National Renewable Energy Laboratory (NREL)

In this tutorial, we examine a cascading hydropower system, consisting of a network of 15 interconnected hydropower stations. We shall assume the presence of three balancing authorities (BAs) - BA1, BA2, and BA3. Each of these authorities will have distinct jurisdictional connections with their local reservoirs.

The jurisdictional connections of the stations, reservoirs, and balancing authorities are illustrated below:



For this tutorial, we assume that, apart from hydropower, no other existing power generation technologies or transmission lines are in place.

However, we will be exploring the potential of incorporating four additional technologies into our grid:

- Coal-fired plants
- Wind power plants
- Solar power plants
- Energy storage plants

The objective of our scenario is to devise an electric mix pathway from 2020 to 2030 that enables the achievement of zero-carbon emissions. We shall use a 48-hour period as a representative sample for our analysis.

4.4 Mathematical Notation

This page provides a detailed description of the mathematical notations used for formulating the model's objective function and constraints (Liu and He, 2023).

4.4.1 Unit List

The description of the units used in this page are as follows:

Unit	Description
hr	Hour
yr	Year
dollar	US Dollar
kW	Kilowatt
MW	Megawatt
MWh	Megawatt-hour
MWy	Megawatt-year
MW-km	Megawatt-kilometer
tonne	Tonne
m	Meter
s	Second
N/A	Not Applicable

4.4.2 Set List

Set	Description	Unit
$e \in \mathcal{E}$	Technology	N/A
$h, h_{\text{start}}, h_{\text{end}} \in \mathcal{H}$	Hour	hr
$y, y_{\text{next}}, y_{\text{pre}}, y_{\text{start}}, y_{\text{end}} \in \mathcal{Y}$	Year	yr
$m \in \mathcal{M}$	Month	N/A
$z, z_{\text{from}}, z_{\text{to}} \in \mathcal{Z}$	Zone	N/A
age $\in \mathcal{AGE}$	Operation time	yr
$s, \text{su} \in \mathcal{S}$	Hydropower station	N/A
$\mathcal{IU}_s \in \mathcal{S}$	Immediate upstream hydropower stations of hydropower station s	N/A
$\mathcal{SZ}_z \in \mathcal{S}$	Subset of hydropower stations located in zone z	N/A
$\mathcal{DISP} \in \mathcal{E}$	Subset of dispatchable technology	N/A
$\mathcal{NDISP} \in \mathcal{E}$	Subset of non-dispatchable technology	N/A
$\mathcal{STOR} \in \mathcal{E}$	Subset of storage technology	N/A

4.4.3 Variable List

Symbol	Description	Unit
$\text{cost}^{\text{total}}$	System-wide total cost.	dollar
$\text{cost}^{\text{var}}_{\text{tech}}$	System-wide variable Operation and Maintenance (O&M) cost of technologies.	dollar
$\text{cost}^{\text{fuel}}$	System-wide fuel cost of technologies.	dollar
$\text{cost}^{\text{fix}}_{\text{tech}}$	System-wide fixed O&M cost of technologies.	dollar
$\text{cost}^{\text{fix}}_{\text{line}}$	System-wide fixed O&M cost of transmission lines.	dollar
$\text{cost}^{\text{inv}}_{\text{tech}}$	System-wide capital cost of technologies.	dollar
$\text{cost}^{\text{inv}}_{\text{line}}$	System-wide capital cost of transmission lines.	dollar
$\text{cost}^{\text{annualfuel}}_{y}$	Fuel cost of technologies in the modelled year:math:y (the present value of modelled year y).	dollar
$\text{cost}^{\text{fuel}}_{y}$	Fuel cost of technologies accumulated from modelled year y to a non-modelled year before the immediate next modelled year (the present value of modelled year y).	dollar
$\text{gen}_{h,m,y,z,e}$	Power generation of technology e in zone z in hour h in month m of year y.	MWh

continues on next page

Table 3 – continued from previous page

Symbol	Description	Unit
$\text{charge}_{h,m,y,z,e}$	Charging electricity of storage technology e in zone z in hour h in month m of year y.	MWh
$\text{export}_{h,m,y,z_{\text{from}},z_{\text{to}}}$	Electric energy exported from zone z_{from} to zone z_{to} in hour h in month m of year y.	MWh
$\text{import}_{h,m,y,z_{\text{from}},z_{\text{to}}}$	Electric energy imported from zone z_{from} to zone z_{to} in hour h in month m of year y.	MWh
$\text{storage}^{\text{energy}}_{h,m,y,z,e}$	Energy storage level of storage technology e in hour h in month m of year y in zone z.	MWh
$\text{storage}^{\text{reservoir}}_{s,h,m,y}$	Reservoir storage corresponding to hydropower station s in hour h in month m of year y.	m^3
$\text{power}_{h,m,y,z,e}$	Overall power output of technology e in zone z in hour h in month m of year y.	MW
$\text{power}^{c}_{h,m,y,z,e}$	Charging power of storage technology e in zone z in hour h in month m of year y.	MW
$\text{power}^{\text{up}}_{h,m,y,z,e}$	Increment in power output of technology e in zone z from hour h-1 to hour h in month m of year y.	MW
$\text{power}^{\text{down}}_{h,m,y,z,e}$	Decrement in power output of technology e in zone z from hour h-1 to hour h in month m of year y.	MW
$\text{power}^{\text{hydro}}_{s,h,m,y}$	Power output of hydropower station s in hour h in month m of year y.	MW
$\text{cap}^{\text{existingtech}}_{y,z,e}$	Existing installed capacity of technology e in year y in zone z.	MW
$\text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}}$	Existing transmission capacity in year y from zone z_{from} to zone z_{to}.	MW
$\text{cap}^{\text{invtech}}_{y,z,e}$	Installed capacity of newly built technology e in year y in zone z.	MW
$\text{cap}^{\text{invline}}_{y,z_{\text{from}},z_{\text{to}}}$	Capacity of newly built transmission lines from zone z_{from} to zone z_{to} in year y	MW
$\text{cap}^{\text{remaining}}_{y,z,e}$	Remaining installed capacity of technology e in year y in zone z	MW
$\text{carbon}^{\text{tech}}_{y,e}$	Carbon dioxide equivalent emissions of technology e in year y	tonne
carbon_{y}	Carbon dioxide equivalent emissions of the entire energy system in year y	tonne
$\text{inflow}^{\text{total}}_{s,h,m,y}$	Total inflow of reservoir corresponding to hydropower station s in hour h in month m of year y	m^3/s
$\text{outflow}^{\text{total}}_{s,h,m,y}$	Total outflow of reservoir corresponding to hydropower station s in hour h in month m of year y	m^3/s
$\text{outflow}^{\text{gen}}_{s,h,m,y}$	Generation outflow of reservoir corresponding to hydropower station s in hour h in month m of year y	m^3/s
$\text{outflow}^{\text{withdraw}}_{s,h,m,y}$	Water withdrawal of reservoir corresponding to hydropower station s in hour h in month m of year y	m^3/s
$\text{outflow}^{\text{spillage}}_{s,h,m,y}$	Spillage outflow of reservoir corresponding to hydropower station s in hour h in month m of year y	m^3/s
$\text{head}^{\text{net}}_{s,h,m,y}$	Net water head of hydropower station s in hour h in month m of year y	m
$\text{head}^{\text{loss}}_{s,h,m,y}$	Water head loss of hydropower station s in hour h in month m of year y	m
$z^{\text{forebay}}_{s,h,m,y}$	Forebay water level of reservoir corresponding to hydropower station s in hour h in month m of year y	m
$z^{\text{tailrace}}_{s,h,m,y}$	Tailrace water level of reservoir corresponding to hydropower station s in hour h in month m of year y	m

4.4.4 Parameter List

Symbol	Description	Unit
$C^{\text{vartech}}_{y,z,e}$	Variable O&M cost per unit power generation from technology e in year y in zone z.	dollar/MWh
$C^{\text{fuel}}_{y,z,e}$	Fuel cost per unit power generation from technology e in year y in zone z.	dollar/MWh
$C^{\text{fixtech}}_{y,z,e}$	Fixed O&M cost per year per unit existing capacity of technology e in year y in zone z.	dollar/MW-yr
$C^{\text{invtech}}_{y,z,e}$	Capital cost per unit installed capacity of technology e in year y in zone z.	dollar/MW
$C^{\text{varline}}_{y,z_{\text{from}},z_{\text{to}}}$	Variable O&M cost per unit transmitted electricity from zone z_{from} to zone z_{to} in year y.	dollar/MWh
$C^{\text{fixline}}_{y,z_{\text{from}},z_{\text{to}}}$	Fixed O&M cost per year per unit existing capacity of transmission line from zone z_{from} to zone z_{to} in year y.	dollar/MW-yr
$C^{\text{invline}}_{y,z_{\text{from}},z_{\text{to}}}$	Capital cost per unit expansion of transmission line from zone z_{from} to zone z_{to} in year y.	dollar/MW
$\text{CARBON}_{y,z,e}$	Carbon dioxide equivalent emission per unit power generation from technology e in year y in zone z.	tonne/MWh
$\overline{\text{CARBON}}_{y}$	Upper bound of carbon dioxide equivalent emission summed across all zones and technologies in year y.	tonne
$\text{DEMAND}_{h,m,y,z}$	Average power demand in hour h in month m of year y in zone z.	MW
$\text{CAP}^{\text{inittech}}_{\text{age},z,e}$	Initial installed capacity of technology e with the operation time of age years in zone z.	N/A
$\text{CAP}^{\text{initline}}_{\text{age},z_{\text{from}},z_{\text{to}}}$	Initial installed capacity of transmission lines with the operation time of age years from zone z_{from} to zone z_{to}.	MW
$\text{CAP}^{\text{hydro}}_{s}$	Nameplate capacity of hydropower station s.	MW
$\underline{\text{POWER}}^{\text{hydro}}_{s}$	Guaranteed minimum power output of hydropower station s.	N/A
$\underline{\text{POWER}}^{c}_{h,m,y,z,e}$	Minimum charge power of storage technology e in hour h in month m of year y in zone z, expressed as a percentage of the existing capacity of storage technology e.	N/A
$\text{STORAGE}^{\text{energy}}_{m,y,z,e}$	Energy storage level of technology e at the beginning of month m of year y in zone z, expressed as a percentage of the maximum energy storage capacity of storage technology e.	N/A
R^{up}_{e}	Allowed maximum ramping up capacity of technology e in two successive periods, expressed as a percentage of the existing capacity of storage technology e.	1/hr
R^{down}_{e}	Allowed maximum ramping down capacity of technology e in two successive periods, expressed as a percentage of the existing capacity of storage technology e.	1/hr
$\text{STORAGE}^{\text{initreservoir}}_{s,m,y}$	Initial reservoir storage corresponding to hydropower station s in month m of year y.	m^3
$\text{STORAGE}^{\text{endreservoir}}_{s,m,y}$	Terminal reservoir storage corresponding to hydropower station s in month m of year y.	m^3
$\overline{\text{STORAGE}}^{\text{reservoir}}_{s}$	Upper bound of reservoir storage corresponding to hydropower station s.	m^3
$\underline{\text{STORAGE}}^{\text{reservoir}}_{s}$	Lower bound of reservoir storage corresponding to hydropower station s.	m^3
$\text{INFLOW}^{\text{net}}_{s,h,m,y}$	Net inflow of reservoir corresponding to hydropower station s in hour h in month m of year y.	m^3/s

continues on next page

Table 4 – continued from previous page

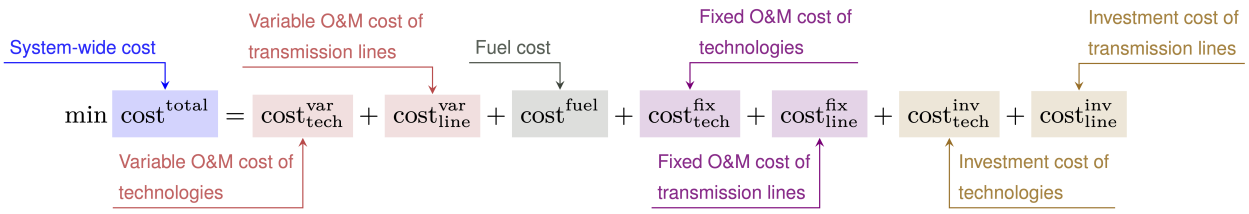
Symbol	Description	Unit
$\mathrm{OUTFLOW}_s^{\mathrm{gen}}$	Maximum outflow that can be released through turbines of hydropower station s.	m^3/s
$\mathrm{OUTFLOW}_s^{\mathrm{spillage}}$	Maximum outflow that can be released through spillway of reservoir corresponding to hydropower station s.	m^3/s
$\mathrm{OUTFLOW}_s$	Minimum outflow of reservoir corresponding to hydropower station s to meet water supply, environmental flow requirements, flood management, and others.	m^3/s
ω	Weight factor to extrapolate representative operation day(s) to a full year (8760 hours).	N/A
ρ	Density of water.	kg/m^3
g	Acceleration of gravity.	m/s^2
$\eta_{y,e}^{\mathrm{in}}$	Charging efficiency of storage technology e in year y.	N/A
$\eta_{y,e}^{\mathrm{out}}$	Generation efficiency of technology e in year y.	N/A
η_s	Generation efficiency of converting water energy to electric energy in hydropower station s.	N/A
$\eta_{z_{\mathrm{from}},z_{\mathrm{to}}}^{\mathrm{trans}}$	Transmission efficiency of transmission lines from zone z_{from} to zone z_{to}.	N/A
$\tau_{\mathrm{su},s}$	Water travel (or propagation) time from the upstream hydropower station su to the immediate downstream hydropower station s.	hr
Δh	Time step.	hr
r	Discount rate.	N/A
T_e	Lifetime of technology e.	yr
T_{line}	Lifetime of transmission line.	yr
EP_e	Power to energy ratio of storage technology e.	hr

4.4.5 Objective Functions

Costs

The objective function of the model is to minimize the net present value of the system's cost. This includes capital cost, fixed O&M cost, variable cost and fuel cost by cost type, technology cost, transmission line cost by the source of cost, and operation cost and planning cost by the source of cost.

The cost equations are defined as follows:



$$\min \text{cost}^{\text{total}} = \text{cost}^{\text{var}}_{\text{tech}} + \text{cost}^{\text{var}}_{\text{line}} + \text{cost}^{\text{fuel}} + \text{cost}^{\text{fix}}_{\text{tech}} + \text{cost}^{\text{fix}}_{\text{line}} + \text{cost}^{\text{inv}}_{\text{tech}} + \text{cost}^{\text{inv}}_{\text{line}}$$

$$\text{cost}^{\text{var}}_{\text{tech}} = \frac{\sum_{h,m,y,z,\text{e}} C^{\text{tech-var}}_{y,z,\text{e}} \times \text{gen}_{\text{h,m,y,z,e}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{var}}_{\text{line}} = \frac{\sum_{h,m,y,z_s,z_o} C^{\text{line-var}}_{y,z} \times \text{export}_{\text{h,m,y,z_s,z_o}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{fuel}} = \frac{\sum_{h,m,y,z,\text{e}} C^{\text{fuel}}_{y,z,\text{e}} \times \text{gen}_{\text{h,m,y,z,e}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{fix}}_{\text{tech}} = \sum_{y,z,\text{e}} C^{\text{tech-fix}}_{y,z,\text{e}} \times \text{cap}^{\text{existing-tech}}_{\text{y,z,e}} \times \text{factor}^{\text{fix}}_{\text{y}}$$

$$\text{cost}^{\text{fix}}_{\text{line}} = \sum_{y,z_s,z_o} C^{\text{line-fix}}_{y,z_s,z_o} \times \text{cap}^{\text{existing-line}}_{\text{y,z_s,z_o}} \times \text{factor}^{\text{fix}}_{\text{y}}$$

$$\text{cost}^{\text{inv}}_{\text{tech}} = \sum_{y,z,\text{e}} C^{\text{tech-inv}}_{y,z,\text{e}} \times \text{cap}^{\text{tech-inv}}_{\text{y,z,e}} \times \text{factor}^{\text{inv}}_{\text{y}}$$

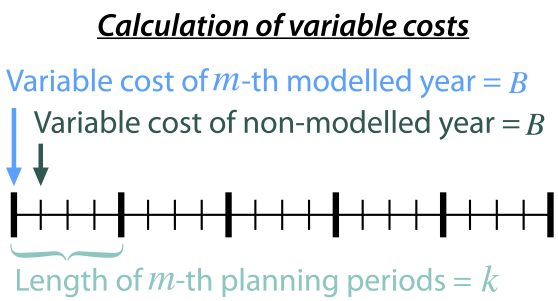
$$\text{cost}^{\text{inv}}_{\text{line}} = \sum_{y,z_s,z_o} C^{\text{line-inv}}_{y,z_s,z_o} \times \text{cap}^{\text{line-inv}}_{\text{y,z_s,z_o}} \times \text{factor}^{\text{inv}}_{\text{y}} \times 0.5$$

Factors

To account for the variable factor, fixed factor, and capital factor, we need to convert all future costs to their net present value. This means adjusting for the time value of money so that all costs are expressed in terms of today's dollars.

We also assume that variable cost and fixed cost for non-modelled years are assumed to be equal to the cost of the last modelled year preceding them. This allows for consistent comparison across different time periods and technologies.

Variable Factor



Given the following:

- Variable cost of modeled year: B
- Discount rate: r
- m-th modeled year: $m = y - y_{\text{min}}$
- Depreciation periods: n

The total present value can be calculated as follows:

$$\text{total present value} = \frac{B}{(1+r)^m} + \frac{B}{(1+r)^{m+1}} + \cdots + \frac{B}{(1+r)^{(m+k-1)}}$$

$$= B(1+r)^{(1-m)} \frac{1-(1+r)^{-k}}{r}$$

And we can calculate the variable factor as follows:

$$\text{factor}_y^{\text{var}} = (1+r)^{1-m_y} \frac{1-(1+r)^{-k_y}}{r}$$

$$m_y = y - y_{\text{min}}$$

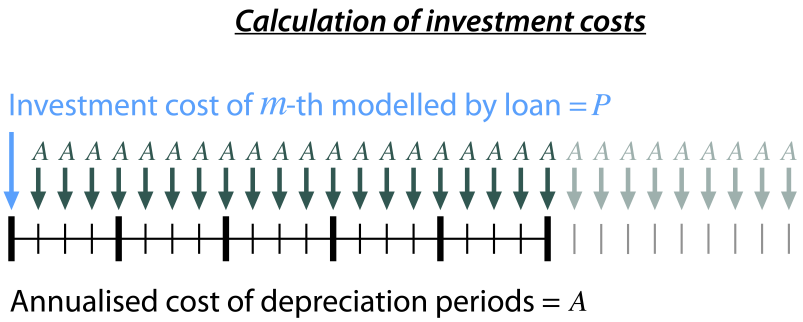
$$k_y = y_{\text{periods}}$$

Fixed Factor

We can equate the fixed factor with the variable factor as follows:

$$\text{factor}_y^{\text{fix}} = \text{factor}_y^{\text{var}}$$

Investment Factor



Given the following:

- Weighted Average Cost of Capital (WACC, or otherwise known as the interest rate): i
- Discount rate: r
- m-th modeled year: $m = y - y_{\text{min}}$
- Length of m-th planning periods: k

The total present value can be calculated as follows:

$$\text{total present value} = \frac{P}{(1+r)^m}$$

$$= \frac{\frac{A}{(1+i)} + \frac{A}{(1+i)^2} + \cdots + \frac{A}{(1+i)^n}}{(1+r)^m}$$

$$= A\frac{1-(1+i)^{-n}}{i} \times \frac{1}{(1+r)^m}$$

From the above, we can solve for the annualized cost of depreciation periods, A, as:

$$A = P\frac{i}{1-(1+i)^{-n}}$$

The capital recovery factor is then calculated as:

$$\text{capital recovery factor} = \frac{i}{1-(1+i)^{-n}}$$

Let's focus on the time periods that fall within the modelled time horizon (indicated in black colour). We can calculate the length of time periods, k, as follows:

$$k = y_{\text{max}} - y$$

Using k, we can calculate the net present value as follows:

$$\text{net present value} = \frac{\frac{A}{(1+r)} + \frac{A}{(1+r)^2} + \cdots + \frac{A}{(1+r)^{\min(n,k)}}}{(1+r)^m} = A \times \frac{1-(1+r)^{-\min(n,k)}}{r(1+r)^m}$$

And we can calculate the investment factor as follows:

$$\text{factor}_y^{\text{inv}} = \frac{i}{1-(1+i)^{-n}} \times \frac{1-(1+r)^{-\min(n,k)}}{r(1+r)^m}$$

4.4.6 Constraints

Retirement

The model computes the retirement of each technology and transmission line with these considerations:

- The historical capacity of the technology and transmission line is based on its capacity ratio.
- Each planning and scheduling period is based on the existing capacity.

The existing capacity for each year, in each zone, for each technology, is as follows:

$$\text{cap}_{y,z,e}^{\text{existingtech}} = \sum_{\text{age}=1}^{T_e-(y-y_{\text{start}})} \text{CAP}_{\text{age},z,e}^{\text{inittech}} + \sum_{y_{\text{pre}}=\max(y_{\text{start}},y-T_e)}^{y} \text{cap}_{y_{\text{pre}},z,e}^{\text{invtech}} \quad \forall y,z,e$$

The existing capacity of the transmission lines for each year, from z_{from} zone to z_{to}-th zone, is as follows:

$$\text{cap}_{y,z_{\text{from}},z_{\text{to}}}^{\text{existingline}} = \sum_{\text{age}=1}^{T_{\text{line}}-(y-y_{\text{start}})} \text{CAP}_{\text{age},z_{\text{from}},z_{\text{to}}}^{\text{initline}} + \sum_{y_{\text{pre}}=\max(y_{\text{start}},y-T_{\text{line}})}^{y} \text{cap}_{y_{\text{pre}},z_{\text{from}},z_{\text{to}}}^{\text{invline}} \quad \forall y, z_{\text{from}} \neq z_{\text{to}}$$

Carbon Emission

The model computes the carbon emissions for each year, based on the sum of carbon emissions from each zone, and from each technology as follows:

$$\text{carbon}_y = \sum_{e \in \mathcal{E}} \sum_{z \in \mathcal{Z}} \sum_{m \in \mathcal{M}} \sum_{h \in \mathcal{H}} \left(\text{CARBON}_{y,z,e} \times \text{gen}_{h,m,y,z,e} \right) \quad \forall y$$

The calculated carbon emission for each year lower than its upper bound, as follows:

$$\text{carbon}_y \leq \overline{\text{CARBON}}_y \quad \forall y$$

Power Balance

The model computes the power balance for each hour, in each time period, for each year, and in each zone, as follows:

$$\text{DEMAND}_{h,m,y,z} \times \Delta h = \sum_{z_{\text{from}} \in \mathcal{Z} \setminus \{z\}} \text{import}_{h,m,y,z_{\text{from}},z} - \sum_{z_{\text{to}} \in \mathcal{Z} \setminus \{z\}} \text{export}_{h,m,y,z,z_{\text{to}}}$$

$$+ \sum_{e \in \mathcal{E}} \text{gen}_{h,m,y,z,e} - \sum_{e \in \mathcal{STOR}} \text{charge}_{h,m,y,z,e} \quad \forall h, m, y, z$$

Transmission

We simplify the transmission of electricity as a transportation model. The model computes the transmission loss for each hour, in each time period, for each year, from z_{from} zone to z_{to} zone, as follows:

$$\text{import}_{h,m,y,z_{\text{from}},z_{\text{to}}} = \text{export}_{h,m,y,z_{\text{from}},z_{\text{to}}} \times \eta^{\text{trans}}_{z_{\text{from}},z_{\text{to}}} \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$

This model assumes that the transmitted power of each transmission line is only constrained by the transmission capacity between two zones as follows:

$$\text{import}_{h,m,y,z_{\text{from}},z_{\text{to}}} \leq \text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}} \times \Delta h \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$

$$\text{export}_{h,m,y,z_{\text{from}},z_{\text{to}}} \leq \text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}} \times \Delta h \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$

Power Output

The power output of storage and each dispatchable (exclude hydropower) technology ($\text{power}_{h,m,y,z,e}$) is limited by the existing installed capacity ($\text{cap}^{\text{existingtech}}_{y,z,e}$) and minimum technical output, as follows:

$$\underline{\text{POWER}}_{h,m,y,z,e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \leq \text{power}_{h,m,y,z,e} \leq \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall h, m, y, z, e \in \mathcal{STOR} \,\&\, \mathcal{DISP}$$

Since hydropower processes are explicitly modelled at the plant level in PREP-SHOT, total hydropower output in zone z ($\text{power}_{h,m,y,z,e=\text{hydro}}$) is the sum of the plant-level hydropower output ($\text{power}^{\text{hydro}}_{s,h,m,y}$):

$$\text{power}_{h,m,y,z,e=\text{hydro}} = \sum_{s \in \mathcal{SZ}_z} \text{power}^{\text{hydro}}_{s,h,m,y} \quad \forall h, m, y, z$$

Here, calculation of $\text{power}_{s,h,m,y}^{\text{hydro}}$ is obtained by external net water head simulation procedure. In addition, $\text{power}_{s,h,m,y}^{\text{hydro}}$ is bounded between the guaranteed minimum output ($\underline{\text{POWER}}_{s}^{\text{hydro}}$) and the nameplate capacity ($\text{CAP}_{s}^{\text{hydro}}$), as follows:

$$\underline{\text{POWER}}_{s}^{\text{hydro}} \leq \text{power}_{s,h,m,y}^{\text{hydro}} \leq \text{CAP}_{s}^{\text{hydro}} \quad \forall s,h,m,y$$

For VRE, their power output is constrained by the capacity factors as follows:

$$\text{power}_{h,m,y,z,e} \leq \text{CF}_{h,m,y,z,e} \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h,m,y,z,e \in \mathcal{NDISP}$$

Regardless of the technology type, actual power generation ($\text{gen}_{h,m,y,z,e}$) in a corresponding period Δh can be calculated based on the power output ($\text{power}_{h,m,y,z,e}$) and the generation efficiency ($\eta_{y,e}^{\text{out}}$):

$$\text{gen}_{h,m,y,z,e} = \text{power}_{h,m,y,z,e} \times \Delta h \times \eta_{y,e}^{\text{out}} \quad \forall h,m,y,z,e \in \mathcal{E}$$

Note that $\eta_{y,e}^{\text{out}} = 1$ when $e \in \mathcal{E} \backslash \mathcal{STOR}$.

Power output variation

All technologies apart from non-dispatchable technology are limited by the so-called ramping capability, meaning that the variation of their power output in two successive periods is limited. We introduce two non-negative auxiliary variables: increment ($\text{power}_{h,m,y,z,e}^{\text{up}}$) and decrement ($\text{power}_{h,m,y,z,e}^{\text{down}}$) to describe changes in power output in two successive periods (from h-1 to h) as follows:

$$\text{power}_{h,m,y,z,e}^{\text{up}} - \text{power}_{h,m,y,z,e}^{\text{down}} = \text{power}_{h,m,y,z,e} - \text{power}_{h-1,m,y,z,e} \quad \forall h,m,y,z,e \in \mathcal{E} \backslash \mathcal{NDISP}$$

When the power plant ramps up from h-1 to h, the minimum of $\text{power}_{h,m,y,z,e}^{\text{up}}$ is obtained when $\text{power}_{h,m,y,z,e}^{\text{down}}$ becomes zero. Similarly, when the power plant ramps down from h-1 to h, the minimum of $\text{power}_{h,m,y,z,e}^{\text{down}}$ is obtained when $\text{power}_{h,m,y,z,e}^{\text{up}}$ becomes zero. Therefore, we can constrain the maximum ramping up and down respectively, as follows:

$$\text{power}_{h,m,y,z,e}^{\text{up}} \leq R_{e}^{\text{up}} \times \Delta h \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h,m,y,z,e \in \mathcal{E} \backslash \mathcal{NDISP}$$

$$\text{power}_{h,m,y,z,e}^{\text{down}} \leq R_{e}^{\text{down}} \times \Delta h \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h,m,y,z,e \in \mathcal{E} \backslash \mathcal{NDISP}$$

where R_{e}^{up}/R_{e}^{down} is the allowed maximum/minimum ramping up/down capacity of technology e in two successive periods, expressed as a percentage of the existing capacity of storage technology e.

Energy storage

Similar to the power discharging process, the charging power of storage technology e ($\text{power}_{h,m,y,z,e}^{c}$) is also limited by the existing installed capacity and technical minimum charging power ($\underline{\text{POWER}}_{h,m,y,z,e}^{c}$) as follows:

$$\underline{\text{POWER}}_{h,m,y,z,e}^{c} \times \text{cap}_{y,z,e}^{\text{existingtech}} \leq \text{power}_{h,m,y,z,e}^{c} \leq \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h,m,y,z,e \in \mathcal{STOR}$$

The charging generation ($\text{charge}_{h,m,y,z,e}$) and $\text{power}^{c}_{h,m,y,z,e}$ need to meet the following formula:

$$\text{charge}_{h,m,y,z,e} = \text{power}^{\text{c}}_{h,m,y,z,e} \times \Delta h \times \eta^{\text{in}}_{y,e} \quad \forall h, m, y, z, e \in \mathcal{STOR}$$

Changes in stored electricity ($\text{storage}^{\text{energy}}_{h,m,y,z,e}$) in two successive periods should be balanced by the charging ($\text{charge}_{h,m,y,z,e}$) and discharging ($\text{gen}_{h,m,y,z,e}$) processes:

$$\text{storage}^{\text{energy}}_{h,m,y,z,e} - \text{storage}^{\text{energy}}_{h-1,m,y,z,e} = \text{charge}_{h,m,y,z,e} - \text{gen}_{h,m,y,z,e}$$

In addition, the initial (when $h = h_{\text{start}}$) stored electricity ($\text{storage}^{\text{energy}}_{h=h_{\text{start}},m,y,z,e}$) of storage technology e in each month of each year can be calculated based on the proportion of the maximum storage capacity, as follows:

$$\text{storage}^{\text{energy}}_{h=h_{\text{start}},m,y,z,e} = \text{STORAGE}^{\text{energy}}_{m,y,z,e} \times \text{EP}_{e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall m, y, z, e \in \mathcal{STOR}$$

The instantaneous storage energy level ($\text{storage}^{\text{energy}}_{h,m,y,z,e}$) of storage technology e should not exceed the maximum energy storage capacity, as follows:

$$\text{storage}^{\text{energy}}_{h,m,y,z,e} \leq \text{EP}_{e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall h, m, y, z, e \in \mathcal{STOR}$$

Water balance

Similar to the storage technologies, changes in reservoir storage ($\text{storage}^{\text{reservoir}}_{s,h,m,y}$) in two successive periods should be balanced by total inflow ($\text{inflow}^{\text{total}}_{s,h,m,y}$) and total outflow ($\text{outflow}^{\text{total}}_{s,h,m,y}$):

$$\text{storage}^{\text{reservoir}}_{s,h,m,y} - \text{storage}^{\text{reservoir}}_{s,h-1,m,y} = \Delta h \times 3600 \times \left(\text{inflow}^{\text{total}}_{s,h,m,y} - \text{outflow}^{\text{total}}_{s,h,m,y}\right) \quad \forall s, h, m, y$$

Here $\text{inflow}^{\text{total}}_{s,h,m,y}$ consists of two parts: the total outflow received from all immediate upstream reservoirs ($\sum_{\text{su} \in \mathcal{IU}_s} \text{outflow}^{\text{total}}_{\text{su},h-\tau_{\text{su},s},m,y}$) and the net inflow (also called incremental inflow) of the drainage area controlled by this hydropower reservoir ($\text{INFLOW}^{\text{net}}_{s,h,m,y}$), which can be expressed as follows:

$$\text{inflow}^{\text{total}}_{s,h,m,y} = \text{INFLOW}^{\text{net}}_{s,h,m,y} + \sum_{\text{su} \in \mathcal{IU}_s} \text{outflow}^{\text{total}}_{\text{su},h-\tau_{\text{su},s},m,y} \quad \forall s, h, m, y$$

Note that PREP-SHOT assumes a constant water travel (or propagation) time ($\tau_{\text{su},s}$). The total outflow of each reservoir consists of three parts: upstream water withdrawal (i.e., water used for non-hydro purposes such as agriculture irrigation and urban water supply) ($\text{outflow}^{\text{withdraw}}_{s,h,m,y}$), generation flow (i.e., water flow through the turbines of the hydropower plant) ($\text{outflow}^{\text{gen}}_{s,h,m,y}$) and spillage flow (i.e., water spilled over the spillways) ($\text{outflow}^{\text{spillage}}_{s,h,m,y}$):

$$\text{outflow}^{\text{total}}_{s,h,m,y} = \text{outflow}^{\text{withdraw}}_{s,h,m,y} + \text{outflow}^{\text{gen}}_{s,h,m,y} + \text{outflow}^{\text{spillage}}_{s,h,m,y} \quad \forall s, h, m, y$$

Reservoir outflow

The generation flow and spillage flow of the reservoir are limited by the maximum outflow capacity of turbines ($\mathrm{OUTFLOW}_s^{\mathrm{gen}}$) and spillway ($\mathrm{OUTFLOW}_s^{\mathrm{spillage}}$), respectively. The sum of these two parts also needs to meet the minimum outflow required ($\mathrm{OUTFLOW}_s$) for other purposes (e.g., ecological flow, shipping flow). These constraints are summarized as:

$$\mathrm{outflow}_{s,h,m,y}^{\mathrm{gen}} \leq \mathrm{OUTFLOW}_s^{\mathrm{gen}} \quad \forall s,h,m,y$$

$$\mathrm{outflow}_{s,h,m,y}^{\mathrm{spillage}} \leq \mathrm{OUTFLOW}_s^{\mathrm{spillage}} \quad \forall s,h,m,y$$

$$\mathrm{OUTFLOW}_s \leq \mathrm{outflow}_{s,h,m,y}^{\mathrm{gen}} + \mathrm{outflow}_{s,h,m,y}^{\mathrm{spillage}} \quad \forall s,h,m,y$$

Reservoir storage

The initial (when $h = h_{\mathrm{start}}$) and terminal (when $h = h_{\mathrm{end}}$) storage ($\mathrm{storage}_{s,h=h_{\mathrm{start}},m,y}^{\mathrm{reservoir}}$ and $\mathrm{storage}_{s,h=h_{\mathrm{end}},m,y}^{\mathrm{reservoir}}$) of hydropower reservoir in each month of each year should be assigned as:

$$\mathrm{storage}_{s,h=h_{\mathrm{start}},m,y}^{\mathrm{reservoir}} = \mathrm{STORAGE}_{s,m,y}^{\mathrm{initreservoir}} \quad \forall s,m,y$$

$$\mathrm{storage}_{s,h=h_{\mathrm{end}},m,y}^{\mathrm{reservoir}} = \mathrm{STORAGE}_{s,m,y}^{\mathrm{endreservoir}} \quad \forall s,m,y$$

The reservoir storage is bounded between the maximum ($\overline{\mathrm{STORAGE}}_s^{\mathrm{reservoir}}$) and minimum storage ($\underline{\mathrm{STORAGE}}_s^{\mathrm{reservoir}}$) depending on the functions (e.g., flood control, recreation, and water supply) of the reservoir:

$$\underline{\mathrm{STORAGE}}_s^{\mathrm{reservoir}} \leq \mathrm{storage}_{s,h,m,y}^{\mathrm{reservoir}} \leq \overline{\mathrm{STORAGE}}_s^{\mathrm{reservoir}} \quad \forall s,h,m,y$$

4.5 Discussion

We encourage our community to actively participate and engage in discussions. The diverse perspectives and feedback from our users are valuable to us and will help us improve the model and its documentation.

Check out the FAQ page for answers to common questions.

4.5.1 GitHub Discussions

Users can join us for public discussions on the Discussions page in our GitHub repository.

- Ask questions or seek clarifications.
- Share ideas or best practices.
- Showcase how you have used the model for your projects and share your experiences.
- Engage in constructive conversations with our community.

4.5.2 Private Queries

If you have specific questions, feedback, or topics that are not suitable for public discussions, you may directly reach out to Zhanwei Liu.

4.6 Contribution

We welcome and appreciate contributions for the RPEP-SHOT library. Here are the steps to contribute:

4.6.1 Development Process

1. **Create an Issue**

   If you find a bug or have an idea for an improvement or new feature, please create an issue.

2. **Fork the Repository**

   You can fork the PREP-SHOT repository on GitHub.

3. **Create a Branch**

   Create a new branch in your forked repository and name the branch according to the feature or fix you're working on.

4. **Commit Changes**

   Make changes in your branch. Once you've made improvements or bug fixes to the project, commit the changes with a meaningful commit message.

5. **Run Tests** To execute all tests, navigate to the root directory of PREP-SHOT and run:

```
python -m unittest discover -s tests
```

To check your code for PEP8 compliance, run:

```
pylint run.py
pylint prepshot
```

6. **Start a Pull Request**

   Open a pull request from your forked repository to the main PREP-SHOT repository. Describe your changes in the pull request.

7. **Code Review**

   Maintainers of the PREP-SHOT project will review your code. They may ask for changes or improvements before the code is merged into the main codebase.

Please ensure that you update tests as necessary when you're contributing code, and follow the coding conventions established in the rest of the project.

4.6.2 Building Documentation

The documentation is built using Sphinx. To build the documentation, you need to install the required packages using the following command:

```
pip install -r docs/requirements.txt
```

You can build the documentation using the following command:

```
cd doc
make clean
make html
```

The documentation will be built in the *doc/build* directory.

4.6.3 Contributing Guidelines

PREP-SHOT is written in Python and follows the PEP8 coding standard. Please ensure that your code follows the PEP8 coding standard. You can use the Pylint tool to check your code for PEP8 compliance.

4.7 Changelog

Here, you'll find notable changes for each version of PREP-SHOT.

4.7.1 Version 0.1.0 - Jun 24, 2024

- PREP-SHOT model is released with basic functionality for energy expansion planning.
- Linear programming optimization model for energy systems with multiple zones.
- Support for solvers such as Gurobi, CPLEX, MOSEK, and GLPK via Pyomo.
- Input and output handling with *pandas* and *Xarray*.

4.7.2 Version 0.1.1 - Jul 11, 2024

Added

- Add an example, expansion of Southeast Asia Mainland power system considering hydropower of Lower Mekong River.
- Update the documentation with a docstring for each function and class.
- Add the Semantic Versioning Specification.

Fixed

Changed

- Support for solvers such as GUROBI (Commercial), COPT (Commercial), MOSEK (Commercial), and HiGHS (Open source) via PyOptInterface.
- Change default solver to HiGHS.
- Change the code comment style to NumPy.
- Change the code style to PEP8.
- Categorize constraint definitions based on type (co2, cost, demand, generation, hydro, investment, nondispatchable, storage, transmission) for better organization.
- Split *rule.py* class into serveral smaller, focused classes according to categorized constraint definitions.
- Simplify model by replacing intermediate constraints with direct expressions.
- Extract new modules *solver.py*, *output_data.py*, and *set_up.py* from *run.py* and *utils.py*.
- Remove *parameters.py* into *set_up.py*.
- Refactor and improve comments and function names for clarity and conciseness.

Deprecated

- Removed dependency on Pyomo due to high memory usage and slow performance for large-scale models. For you reference.

4.7.3 Version 0.1.2 - Jul 22, 2024

Added

- Added mathematical notations to the constraint module.
- Added a test script for *prepshot.utils*.

Fixed

- Fixed the format of the API reference.
- Fix code blocks of documentation.
- Updated *Contribution.rst* to include context on running tests and code style checks.
- Defined explicit data types for inputs and outputs of functions for better type checking and readability.
- Added *pyoptinterface._src.core_ext* to Pylint's extension package allow list to resolve cpp-extension-no-member warning.

Changed

- Updated *model.py* to keep necessary decision variables and use expressions for intermediate variables instead of direct determination.
- Refactored *extract_results_non_hydro* in *output_data.py* to extract common features for different variables, simplifying the code.
- Removed definitions of complex sets and opted for simple sets wherever possible to streamline the code.
- Refactor: Organize import order of modules according to PEP 8 guidelines: (1) Grouped standard library imports at the top; (2) Followed by third-party library imports; (3) Local application/library imports at the bottom.

4.8 API Reference

Description: PREP-SHOT package metadata.

4.8.1 set_up

This module contains functions for setting parameters and configuring logging.

prepshot.set_up.**initialize_environment**(*config_files*)

Load configuration data, set up logging, and process input parameters.

Parameters
config_files (*Dict[str, str]*) -- Dictionary containing paths to parameters and configuration files.

Returns
Dictionary containing the global parameters.

Return type
Dict[str, Any]

prepshot.set_up.**parse_cli_arguments**(*params_list*)

Parse command-line arguments from a list of names of parameters.

Parameters
params_list (*List[str]*) -- List of parameters.

Returns
Parsed command-line arguments.

Return type
argparse.Namespace

4.8.2 logs

This module contains functions to set up logging for the model run and to log the start and end of functions.

prepshot.logs.**log_parameter_info**(*config_data*)

Log key parameters used for the model.

Parameters
config_data (*dict*) -- Dictionary containing configuration data for the model.

Return type
None

prepshot.logs.**setup_logging**()

Set up logging configuration for the model run.

Return type
None

prepshot.logs.**timer**(*func*)

Decorator to log the start and end of a function and its runtime.

Parameters
func (*function*) -- The function to be decorated.

Returns
The return value of decorated function.

Return type
Any

4.8.3 load_data

This module contains functions for loading and processing data from JSON and Excel files.

prepshot.load_data.**compute_cost_factors**(*data_store*)

Calculate cost factors for various transmission investment and operational costs.

Parameters
data_store (*dict*) -- Dictionary containing loaded parameters.

Return type
None

prepshot.load_data.**extract_config_data**(*config_data*)

Extract necessary data from configuration settings.

Parameters
config_data (*dict*) -- Configuration data for the model.

Returns
Dictionary containing necessary configuration data.

Return type
dict

prepshot.load_data.**extract_sets**(*data_store*)

Extract simple sets from loaded parameters.

Parameters
data_store (*dict*) -- Dictionary containing loaded parameters.

Return type
None

prepshot.load_data.**load_excel_data**(*input_folder*, *params_info*, *data_store*)

Load data from Excel files based on the provided parameters.

Parameters

- **input_folder** (*str*) -- Path to the input folder.
- **params_info** (*dict*) -- Dictionary containing parameter names and their corresponding file information.
- **data_store** (*dict*) -- Dictionary to store loaded data.

Return type
None

prepshot.load_data.**load_json**(*file_path*)

Load data from a JSON file.

Parameters
file_path (*str*) -- Path to the JSON file.

Returns
Dictionary containing data from the JSON file.

Return type
dict

prepshot.load_data.**process_data**(*params_info*, *input_folder*)

Load and process data from input folder based on parameters settings.

Parameters

- **params_info** (*dict*) -- Dictionary containing parameters information.
- **input_folder** (*str*) -- Path to the input folder.

Returns
Dictionary containing processed parameters.

Return type
dict

prepshot.load_data.**read_excel**(*filename*, *index_cols*, *header_rows*, *unstack_levels=None*, *first_col_only=False*, *dropna=True*)

Read data from an Excel file into a pandas DataFrame.

Parameters

- **filename** (*str*) -- The name of the input Excel file.
- **index_cols** (*list*) -- List of column names to be used as index.
- **header_rows** (*list*) -- List of rows to be used as header.
- **unstack_levels** (*list, optional*) -- List of levels to be unstacked, by default None
- **first_col_only** (*bool, optional*) -- Whether to keep only the first column, by default False
- **dropna** (*bool, optional*) -- Whether to drop rows with NaN values, by default True

Returns
A DataFrame containing the data from the Excel file.

Return type
pandas.DataFrame

4.8.4 solver

This module contains the definition of the solver class.

prepshot.solver.**get_solver**(*params*)

Retrieve the solver object based on parameters.

Parameters
params (*dict*) -- Configuration dictionary with solver details.

Returns
Solver object based on the configuration.

Return type
object

prepshot.solver.**set_solver_parameters**(*model*)

Set the solver-specific parameters for the model.

Parameters
model (*object*) -- Model to configurable.

Return type
None

prepshot.solver.**solve_model**(*model*, *params*)

Solve the model using the provided parameters.

Parameters

- **model** (*object*) -- Model to solve.
- **params** (*dict*) -- Configuration parameters for solving the model.

Returns
True if the model is solved optimally, False otherwise.

Return type
bool

4.8.5 model

This module defines the PREP-SHOT model. The model is created using the pyoptinterface library.

prepshot.model.**create_model**(*params*)

Create the PREP-SHOT model.

Parameters
params (*dict*) -- Dictionary of parameters for the model.

Returns
Model object.

Return type
object

prepshot.model.**define_basic_sets**(*model*)

Define sets for the model.

Parameters
model (*object*) -- Model object to be solved.

Return type
None

prepshot.model.**define_complex_sets**(*model*)

Create complex sets based on simple sets and some conditations. The existing capacity between two zones is set to empty (i.e., No value is filled in the Excel cell), which means that these two zones cannot have newly built transmission lines. If you want to enable two zones which do not have any existing transmission lines, to build new transmission lines in the planning horizon, you need to set their capacity as zero explicitly.

Parameters
model (*object*) -- Model to be solved.

Return type
None

prepshot.model.**define_constraints**(*model*)

Define constraints for the model.

Parameters
model (*object*) -- Model to be solved.

Return type
None

prepshot.model.**define_model**(*params*)

This function creates the model class depending on predefined solver.

Parameters
params (*dict*) -- parameters for the model

Returns
A pyoptinterface Model object depending on the solver

Return type
object

Raises
ValueError -- Unsupported or undefined solver

prepshot.model.**define_variables**(*model*)

Define variables for the model.

Parameters
model (*object*) -- Model to be solved.

Return type
None

4.8.6 constraints

co2

This module contains the constraints and expressions related to carbon emissions. The model computes the carbon emissions for each year, based on the sum of carbon emissions from each zone, and from each technology as follows:

$$\text{carbon}_y = \sum_{e \in \mathcal{E}} \sum_{z \in \mathcal{Z}} \sum_{m \in \mathcal{M}} \sum_{h \in \mathcal{H}} \left(\text{CARBON}_{y,z,e} \times \text{gen}_{h,m,y,z,e} \right) \quad \forall y$$

The calculated carbon emission for each year lower than its upper bound, as follows:

$$\text{carbon}_y \leq \overline{\text{CARBON}}_y \quad \forall y$$

class prepshot._model.co2.**AddCo2EmissionConstraints**(*model*)

Bases: object

Class for carbon emission constraints and calculations.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

carbon_breakdown(*y*, *z*, *te*)

Carbon emission cost breakdown.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The expression of the model.

Return type
poi.ExprBuilder

emission_calc_by_zone_rule(*y*, *z*)

Calculation of annual carbon emissions by zone.

Parameters

- **y** (*int*) -- Planned year.
- **z** (*str*) -- Zone.

Returns
A constraint of the model.

Return type
poi.ConstraintIndex

emission_calc_rule(*y*)

Calculation of annual carbon emission across all zones and technologies.

Parameters
y (*int*) -- Planned year.

Returns
A constraint of the model.

Return type
poi.ConstraintIndex

emission_limit_rule(*y*)

Annual carbon emission limits across all zones and technologies.

Parameters
y (*int*) -- Planned year.

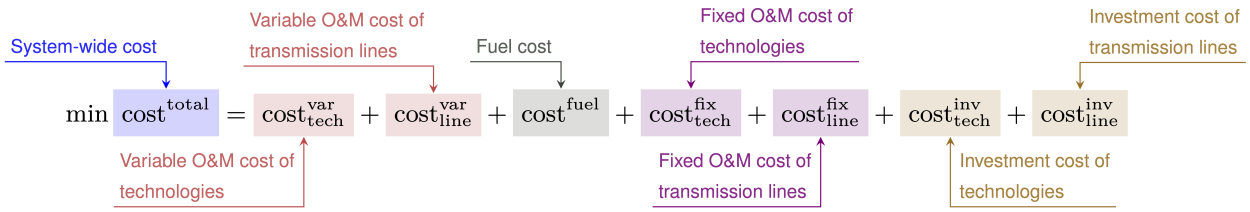
Returns
A constraint of the model.

Return type
poi.ConstraintIndex

cost

This module contains objective functions for the model. The objective function of the model is to minimize the net present value of the system's cost. This includes capital cost, fixed O&M cost, variable cost and fuel cost by cost type, technology cost, transmission line cost by the source of cost, and operation cost and planning cost by the source of cost.

The cost equations are defined as follows:



$$\text{cost}^{\text{var}}_{\text{tech}} = \frac{\sum_{h,m,y,z,\text{e}} C^{\text{tech-var}}_{y,z,\text{e}} \times \text{gen}_{\text{h,m,y,z,e}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{var}}_{\text{line}} = \frac{\sum_{h,m,y,z_s,z_o} C^{\text{line-var}}_{y,z} \times \text{export}_{\text{h,m,y,z_s,z_o}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{fuel}} = \frac{\sum_{h,m,y,z,\text{e}} C^{\text{fuel}}_{y,z,\text{e}} \times \text{gen}_{\text{h,m,y,z,e}}}{\omega} \times \text{factor}^{\text{var}}_{\text{y}}$$

$$\text{cost}^{\text{fix}}_{\text{tech}} = \sum_{y,z,\text{e}} C^{\text{tech-fix}}_{y,z,\text{e}} \times \text{cap}^{\text{existing-tech}}_{\text{y,z,e}} \times \text{factor}^{\text{fix}}_{\text{y}}$$

$$\text{cost}^{\text{fix}}_{\text{line}} = \sum_{y,z_s,z_o} C^{\text{line-fix}}_{y,z_s,z_o} \times \text{cap}^{\text{existing-line}}_{\text{y,z_s,z_o}} \times \text{factor}^{\text{fix}}_{\text{y}}$$

$$\text{cost}^{\text{inv}}_{\text{tech}} = \sum_{y,z,\text{e}} C^{\text{tech-inv}}_{y,z,\text{e}} \times \text{cap}^{\text{tech-inv}}_{\text{y,z,e}} \times \text{factor}^{\text{inv}}_{\text{y}}$$

$$\text{cost}^{\text{inv}}_{\text{line}} = \sum_{y,z_s,z_o} C^{\text{line-inv}}_{y,z_s,z_o} \times \text{cap}^{\text{line-inv}}_{\text{y,z_s,z_o}} \times \text{factor}^{\text{inv}}_{\text{y}} \times 0.5$$

class prepshot._model.cost.**AddCostObjective**(*model*)

Bases: object

Objective function class to determine the total cost of the model.

Parameters
model (*object*) --

__init__(*model*)

The constructor for objective functions class.

Parameters
model (*object*) -- Model to be solved.

Return type
None

cost_fix_line_breakdown(*y*, *z*, *z1*)

Fixed operation and maintenance cost breakdown of transmission lines.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
Fixed operation and maintenance cost of transmission lines at a given year, source and destination zone.

Return type
poi.ExprBuilder

cost_fix_tech_breakdown(*y*, *z*, *te*)

Fixed operation and maintenance cost breakdown.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
Fixed operation and maintenance cost of technologies at a given year, zone and technology.

Return type
poi.ExprBuilder

cost_newline_breakdown(*y*, *z*, *z1*)

New transmission line investment cost breakdown.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
Investment cost of new transmission lines at a given year, source and destination zone.

Return type
poi.ExprBuilder

`cost_newtech_breakdown`(*y*, *z*, *te*)

New technology investment cost breakdown.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
Investment cost of new technologies at a given year, zone and technology.

Return type
poi.ExprBuilder

`cost_var_line_breakdown`(*y*, *z*, *z1*)

Variable operation and maintenance cost breakdown of transmission lines.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
Variable operation and maintenance cost of transmission lines at a given year, source and destination zone.

Return type
poi.ExprBuilder

`cost_var_tech_breakdown`(*y*, *z*, *te*)

Variable operation and maintenance cost breakdown.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
Variable operation and maintenance cost of technologies at a given year, zone and technology.

Return type
poi.ExprBuilder

`define_objective`()

Objective function of the model, to minimize total cost.

Return type
None

`fix_cost_rule`()

Fixed O&M cost of technologies and transmission lines.

Returns
Total fixed O&M cost of technologies and transmission lines over all years, zones and technologies.

Return type
poi.ExprBuilder

`fuel_cost_breakdown`(*y*, *z*, *te*)

Fuel cost breakdown of technologies.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
Fuel cost at a given year, zone and technology.

Return type
poi.ExprBuilder

`income_rule`()

Income from water withdrawal. Reference: https://www.nature.com/articles/s44221-023-00126-0

Returns
Income from water withdrawal.

Return type
poi.ExprBuilder

`newline_cost_rule`()

Total investment cost of new transmission lines.

Returns
Total investment cost of new transmission lines over all years, zones.

Return type
poi.ExprBuilder

`newtech_cost_rule`()

Total investment cost of new technologies.

Returns
Total investment cost of new technologies over all years, zones and technologies.

Return type
poi.ExprBuilder

`var_cost_rule`()

Calculate total variable cost, which is sum of the fuel cost of technologies and variable Operation and maintenance (O&M) cost of technologies and transmission lines.

Returns
Total variable cost across all years, zones and technologies.

Return type
poi.ExprBuilder

demand

This module contains constraints related to demand. The model computes the power balance for each hour, in each time period, for each year, and in each zone, as follows:

$$\begin{aligned}\mathrm{DEMAND}_{h,m,y,z} \times \Delta h = & \sum_{z_{\mathrm{from}} \in \mathcal{Z} \setminus \{z\}} \mathrm{import}_{h,m,y,z_{\mathrm{from}},z} - \sum_{z_{\mathrm{to}} \in \mathcal{Z} \setminus \{z\}} \mathrm{export}_{h,m,y,z,z_{\mathrm{to}}} \\ & + \sum_{e \in \mathcal{E}} \mathrm{gen}_{h,m,y,z,e} - \sum_{e \in \mathcal{STOR}} \mathrm{charge}_{h,m,y,z,e} \quad \forall h, m, y, z\end{aligned}$$

class prepshot._model.demand.**AddDemandConstraints**(*model*)

Bases: object

This class contains demand constraints.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

power_balance_rule(*h*, *m*, *y*, *z*)

Nodal power balance. The total electricity demand for each time period and in each zone should be met by the following.

1. The sum of imported power energy from other zones.
2. The generation from zone z minus the sum of exported power energy from zone z to other zones.
3. The charging power energy of storage technologies in zone z.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

generation

This module contains constraints related to technology generation. The power output of storage and each dispatchable (exclude hydropower) technology ($\text{power}_{h,m,y,z,e}$) is limited by the existing installed capacity ($\text{cap}_{y,z,e}^{\text{existingtech}}$) and minimum technical output, as follows:

$$\underline{\text{POWER}}_{h,m,y,z,e} \times \text{cap}_{y,z,e}^{\text{existingtech}} \leq \text{power}_{h,m,y,z,e} \leq \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h, m, y, z, e \in \mathcal{STOR} \,\&\, \mathcal{DISP}$$

Since hydropower processes are explicitly modelled at the plant level in PREP-SHOT, total hydropower output in zone z ($\text{power}_{h,m,y,z,e=\text{hydro}}$) is the sum of the plant-level hydropower output ($\text{power}_{s,h,m,y}^{\text{hydro}}$):

$$\text{power}_{h,m,y,z,e=\text{hydro}} = \sum_{s \in \mathcal{SZ}_z} \text{power}_{s,h,m,y}^{\text{hydro}} \quad \forall h, m, y, z$$

Here, calculation of $\text{power}_{s,h,m,y}^{\text{hydro}}$ is obtained by external net water head simulation procedure. In addition, $\text{power}_{s,h,m,y}^{\text{hydro}}$ is bounded between the guaranteed minimum output ($\underline{\text{POWER}}_{s}^{\text{hydro}}$) and the nameplate capacity ($\text{CAP}_{s}^{\text{hydro}}$), as follows:

Regardless of the technology type, actual power generation ($\text{gen}_{h,m,y,z,e}$) in a corresponding period Δh can be calculated based on the power output ($\text{power}_{h,m,y,z,e}$) and the generation efficiency ($\eta_{y,e}^{\text{out}}$):

$$\text{gen}_{h,m,y,z,e} = \text{power}_{h,m,y,z,e} \times \Delta h \times \eta_{y,e}^{\text{out}} \quad \forall h, m, y, z, e \in \mathcal{E}$$

Note that $\eta_{y,e}^{\text{out}} = 1$ when $e \in \mathcal{E} \backslash \mathcal{STOR}$.

All technologies apart from dispatchable technology are limited by the so-called ramping capability, meaning that the variation of their power output in two successive periods is limited. We introduce two non-negative auxiliary variables: increment ($mpower_{h,m,y,z,e}^{mup}$) and decrement ($mpower_{h,m,y,z,e}^{mdown}$) to describe changes in power output in two successive periods (from h-1 to h) as follows:

$$\text{power}_{h,m,y,z,e}^{\text{up}} - \text{power}_{h,m,y,z,e}^{\text{down}} = \text{power}_{h,m,y,z,e} - \text{power}_{h-1,m,y,z,e} \quad \forall h, m, y, z, e \ \in \mathcal{E} \backslash \mathcal{NDISP}$$

When the power plant ramps up from h-1 to h, the minimum of $\text{power}_{h,m,y,z,e}^{\text{up}}$ is obtained when $\text{power}_{h,m,y,z,e}^{\text{down}}$ becomes zero. Similarly, when the power plant ramps down from h-1 to h, the minimum of $\text{power}_{h,m,y,z,e}^{\text{down}}$ is obtained when $\text{power}_{h,m,y,z,e}^{\text{up}}$ becomes zero. Therefore, we can constrain the maximum ramping up and down respectively, as follows:

$$\text{power}_{h,m,y,z,e}^{\text{up}} \leq R_{e}^{\text{up}} \times \Delta h \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h, m, y, z, e \in \mathcal{E} \backslash \mathcal{NDISP}$$

$$\text{power}_{h,m,y,z,e}^{\text{down}} \leq R_{e}^{\text{down}} \times \Delta h \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h, m, y, z, e \in \mathcal{E} \backslash \mathcal{NDISP}$$

where $R_{e}^{\text{up}}/R_{e}^{\text{down}}$ is the allowed maximum/minimum ramping up/down capacity of technology e in two successive periods, expressed as a percentage of the existing capacity of storage technology e.

class prepshot._model.generation.**AddGenerationConstraints**(*model*)

Bases: object

Add constraints for generation in the model.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

`gen_up_bound_rule`(*h*, *m*, *y*, *z*, *te*)

Generation is less than or equal to the existing capacity.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`ramping_down_rule`(*h*, *m*, *y*, *z*, *te*)

Ramping down limits.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`ramping_up_rule`(*h*, *m*, *y*, *z*, *te*)

Ramping up limits.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

head_iteration

This module contains utility functions related to the head iteration.

prepshot._model.head_iteration.**compute_error**(*old_waterhead*, *new_waterhead*)

Calculate the error of the water head.

Parameters

- **old_waterhead** (*pandas.DataFrame*) -- The water head before the solution.
- **new_waterhead** (*pandas.DataFrame*) -- The water head after the solution.

Returns
The error of the water head.

Return type
float

prepshot._model.head_iteration.**initialize_waterhead**(*stations*, *year*, *month*, *hour*, *params*)

Initialize water head.

Parameters

- **stations** (*List[str]*) -- List of stations.
- **year** (*List[int]*) -- List of years.
- **month** (*List[int]*) -- List of months.
- **hour** (*List[int]*) -- List of hours.
- **params** (*Dict[str, Any]*) -- Dictionary of parameters for the model.

Returns
A tuple of two pandas.DataFrame objects, the first one is the old water head, the second one is the new water head.

Return type
Tuple[pd.DataFrame, pd.DataFrame]

prepshot._model.head_iteration.**process_model_solution**(*model*, *stations*, *year*, *month*, *hour*, *params*, *old_waterhead*, *new_waterhead*)

Process the solution of the model, updating the water head data.

Parameters

- **model** (*object*) -- Model to be solved.
- **stations** (*list*) -- List f hydropower stations.
- **year** (*list*) -- List of years.
- **month** (*list*) -- List of months.
- **hour** (*list*) -- List of hours.
- **params** (*dict*) -- Dictionary of parameters for the model.
- **old_waterhead** (*pandas.DataFrame*) -- The water head before the solution.
- **new_waterhead** (*pandas.DataFrame*) -- The water head after the solution.

Returns
True if the model is solved, False otherwise.

Return type
bool

prepshot._model.head_iteration.**run_model_iteration**(*model*, *params*, *error_threshold=0.001*, *max_iterations=5*)

Run the model iteratively.

Parameters

- **model** (*object*) -- Model to be solved.
- **params** (*dict*) -- Dictionary of parameters for the model.
- **error_threshold** (*float, optional*) -- The error threshold, by default 0.001
- **max_iterations** (*int, optional*) -- The maximum number of iterations, by default 5

Returns
True if the model is solved, False otherwise.

Return type
bool

hydro

This module contains functions related to hydropower technologies.

1. Water balance of reservoirs.

Similar to the storage technologies, changes in reservoir storage ($\text{storage}^{\text{reservoir}}_{s,h,m,y}$) in two successive periods should be balanced by total inflow ($\text{inflow}^{\text{total}}_{s,h,m,y}$) and total outflow ($\text{outflow}^{\text{total}}_{s,h,m,y}$):

$$\text{storage}^{\text{reservoir}}_{s,h,m,y} - \text{storage}^{\text{reservoir}}_{s,h-1,m,y} = \Delta h \times 3600 \times \left(\text{inflow}^{\text{total}}_{s,h,m,y} - \text{outflow}^{\text{total}}_{s,h,m,y}\right) \quad \forall s,h,m,y$$

Here $\text{inflow}^{\text{total}}_{s,h,m,y}$ consists of two parts: the total outflow received from all immediate upstream reservoirs ($\sum_{\text{su}\in\mathcal{IU}_s} \text{outflow}^{\text{total}}_{\text{su},h-\tau_{\text{su},s},m,y}$) and the net inflow (also called incremental inflow) of the drainage area controlled by this hydropower reservoir ($\text{INFLOW}^{\text{net}}_{s,h,m,y}$), which can be expressed as follows:

$$\text{inflow}^{\text{total}}_{s,h,m,y} = \text{INFLOW}^{\text{net}}_{s,h,m,y} + \sum_{\text{su}\in\mathcal{IU}_s} \text{outflow}^{\text{total}}_{\text{su},h-\tau_{\text{su},s},m,y} \quad \forall s,h,m,y$$

Note that PREP-SHOT assumes a constant water travel (or propagation) time ($\tau_{\text{su},s}$). The total outflow of each reservoir consists of three parts: upstream water withdrawal (i.e., water used for non-hydro purposes such as agriculture irrigation and urban water supply) ($\text{outflow}^{\text{withdraw}}_{s,h,m,y}$), generation flow (i.e., water flow through the turbines of the hydropower plant) ($\text{outflow}^{\text{gen}}_{s,h,m,y}$) and spillage flow (i.e., water spilled over the spillways) ($\text{outflow}^{\text{spillage}}_{s,h,m,y}$):

$$\text{outflow}^{\text{total}}_{s,h,m,y} = \text{outflow}^{\text{withdraw}}_{s,h,m,y} + \text{outflow}^{\text{gen}}_{s,h,m,y} + \text{outflow}^{\text{spillage}}_{s,h,m,y} \quad \forall s,h,m,y$$

2. Reservoir outflow

The generation flow and spillage flow of the reservoir are limited by the maximum outflow capacity of turbines ($\text{OUTFLOW}^{\text{gen}}_{s}$) and spillway ($\text{OUTFLOW}^{\text{spillage}}_{s}$), respectively. The sum of these two parts also needs to meet the minimum outflow required (OUTFLOW_s) for other purposes (e.g., ecological flow, shipping flow). These constraints are summarized as:

$$\text{outflow}^{\text{gen}}_{s,h,m,y} \le \text{OUTFLOW}^{\text{gen}}_{s} \quad \forall s,h,m,y$$

$$\text{outflow}^{\text{spillage}}_{s,h,m,y} \le \text{OUTFLOW}^{\text{spillage}}_{s} \quad \forall s,h,m,y$$

$$\text{OUTFLOW}_s \le \text{outflow}^{\text{gen}}_{s,h,m,y} + \text{outflow}^{\text{spillage}}_{s,h,m,y} \quad \forall s,h,m,y$$

3. Reservoir storage

The initial (when $h = h_{\text{start}}$) and terminal (when $h = h_{\text{end}}$) storage ($\text{storage}^{\text{reservoir}}_{s,h=h_{\text{start}},m,y}$ and $\text{storage}^{\text{reservoir}}_{s,h=h_{\text{end}},m,y}$) of hydropower reservoir in each month of each year should be assigned as:

$$\text{storage}^{\text{reservoir}}_{s,h=h_{\text{start}},m,y} = \text{STORAGE}^{\text{initreservoir}}_{s,m,y} \quad \forall s, m, y$$
$$\text{storage}^{\text{reservoir}}_{s,h=h_{\text{end}},m,y} = \text{STORAGE}^{\text{endreservoir}}_{s,m,y} \quad \forall s, m, y$$

The reservoir storage is bounded between the maximum ($\overline{\text{STORAGE}}^{\text{reservoir}}_{s}$) and minimum storage ($\underline{\text{STORAGE}}^{\text{reservoir}}_{s}$) depending on the functions (e.g., flood control, recreation, and water supply) of the reservoir:

$$\underline{\text{STORAGE}}^{\text{reservoir}}_{s} \leq \text{storage}^{\text{reservoir}}_{s,h,m,y} \leq \overline{\text{STORAGE}}^{\text{reservoir}}_{s} \quad \forall s, h, m, y$$

class prepshot._model.hydro.**AddHydropowerConstraints**(*model*)

Bases: object

Class for hydropower constraints and calculations.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class. Here I define the variables needed and the constraints for the hydropower model.

Parameters
model (*object*) -- Model container which is a dict-like objective and includes parameters, variables and constraints.

Return type
None

end_storage_rule(*s*, *m*, *y*)

Determine storage of reservoir in the terminal hour of each month.

Parameters

- **s** (*str*) -- hydropower plant.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

genflow_up_bound_rule(*s*, *h*, *m*, *y*)

Upper bound of generation flow.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`hydro_output_rule`(*h*, *m*, *y*, *z*)

Hydropower output of all hydropower plants across each zone.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`inflow_rule`(*s*, *h*, *m*, *y*)

Define hydrolic connnect between cascade reservoirs, total inflow of downsteam reservoir = natural inflow + upstream outflow from upsteam reservoir(s).

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
Total inflow of reservoir.

Return type
poi.ExprBuilder

`init_storage_rule`(*s*, *m*, *y*)

Determine storage of reservoir in the initial hour of each month.

Parameters

- **s** (*str*) -- hydropower plant.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`outflow_low_bound_rule`(*s*, *h*, *m*, *y*)

Lower bound of total outflow.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

outflow_rule(*s*, *h*, *m*, *y*)

Total outflow of reservoir is equal to the sum of generation and spillage.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
Total outflow of reservoir.

Return type
poi.ExprBuilder

outflow_up_bound_rule(*s*, *h*, *m*, *y*)

Upper bound of total outflow.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

output_calc_rule(*s*, *h*, *m*, *y*)

Hydropower production calculation. Head parameter is specified after building the model.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`output_low_bound_rule`(*s*, *h*, *m*, *y*)

Lower bound of hydropower output.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`output_up_bound_rule`(*s*, *h*, *m*, *y*)

Upper bound of hydropower output.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`storage_low_bound_rule`(*s*, *h*, *m*, *y*)

Lower bound of reservoir storage.

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`storage_up_bound_rule`(*s*, *h*, *m*, *y*)

Upper bound of reservoir storage.

Parameters

- **s** (*str*) -- hydropower plant.

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

water_balance_rule(*s*, *h*, *m*, *y*)

Water balance of reservoir, i.e., storage[t] = storage[t-1] + net_storage[t].

Parameters

- **s** (*str*) -- hydropower plant.
- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

investment

This module is used to determine investment-related constraints. The model computes the retirement of each technology and transmission line with these considerations:

- The historical capacity of the technology and transmission line is based on its capacity ratio.
- Each planning and scheduling period is based on the existing capacity.

The existing capacity for each year, in each zone, for each technology, is as follows:

$$\text{cap}^{\text{existingtech}}_{y,z,e} = \sum_{\text{age}=1}^{T_e-(y-y_{\text{start}})} \text{CAP}^{\text{inittech}}_{\text{age},z,e} + \sum_{y_{\text{pre}}=\max(y_{\text{start}},y-T_e)}^{y} \text{cap}^{\text{invtech}}_{y_{\text{pre}},z,e} \quad \forall y, z, e$$

The existing capacity of the transmission lines for each year, from z_{from} zone to z_{to}-th zone, is as follows:

$$\text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}} = \sum_{\text{age}=1}^{T_{\text{line}}-(y-y_{\text{start}})} \text{CAP}^{\text{initline}}_{\text{age},z_{\text{from}},z_{\text{to}}} + \sum_{y_{\text{pre}}=\max(y_{\text{start}},y-T_{\text{line}})}^{y} \text{cap}^{\text{invline}}_{y_{\text{pre}},z_{\text{from}},z_{\text{to}}} \quad \forall y, z_{\text{from}} \neq z_{\text{to}}$$

class prepshot._model.investment.**AddInvestmentConstraints**(*model*)

Bases: object

Add constraints for investment in the model.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

new_tech_low_bound_rule(*y*, *z*, *te*)

New investment technology lower bound.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

new_tech_up_bound_rule(*y*, *z*, *te*)

New investment technology upper bound in specific year and zone.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

remaining_capacity_rule(*y*, *z*, *te*)

Remaining capacity of initial technology due to lifetime restrictions. Where in modeled year y, the available technology consists of the following.

1. The remaining in-service installed capacity from the initial technology.
2. The remaining in-service installed capacity from newly built technology in the previous modelled years.

Parameters

- **y** (*int*) -- Planned year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- technology.

Returns
The expression of the model.

Return type
poi.ExprBuilder

tech_lifetime_rule(*y*, *z*, *te*)

Caculation of remaining technology capacity based on lifetime constraints.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The expression of the model.

Return type
poi.ExprBuilder

tech_up_bound_rule(*y*, *z*, *te*)

Allowed capacity of commercial operation technology is less than or equal to the predefined upper bound.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

nondispatchable

This module contains functions related to nondispatchable technologies. For non-dispatchable technologies, their power output is constrained by the predefined capacity factors as follows:

$$\text{power}_{h,m,y,z,e} \leq \text{CF}_{h,m,y,z,e} \times \text{cap}_{y,z,e}^{\text{existingtech}} \quad \forall h, m, y, z, e \in \mathcal{NDISP}$$

class prepshot._model.nondispatchable.**AddNondispatchableConstraints**(*model*)

Bases: object

Add constraints for nondispatchable technologies.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

renew_gen_rule(*h*, *m*, *y*, *z*, *te*)

Renewable generation is determined by the capacity factor and existing capacity.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

storage

This module contains energy storage related functions. Symmetrical energy storage system is considered in this module. It means that the energy storage system has the same power capacity for charging and discharging.

Similar to the power discharging process, the charging power of storage technology e ($\text{power}^{c}_{h,m,y,z,e}$) is also limited by the existing installed capacity and technical minimum charging power ($\underline{\text{POWER}}^{c}_{h,m,y,z,e}$) as follows:

$$\underline{\text{POWER}}^{c}_{h,m,y,z,e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \leq \text{power}^{c}_{h,m,y,z,e} \leq \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall h, m, y, z, e \in \mathcal{STOR}$$

The charging generation ($\text{charge}_{h,m,y,z,e}$) and $\text{power}^{c}_{h,m,y,z,e}$ need to meet the following formula:

$$\text{charge}_{h,m,y,z,e} = \text{power}^{\text{c}}_{h,m,y,z,e} \times \Delta h \times \eta^{\text{in}}_{y,e} \quad \forall h, m, y, z, e \in \mathcal{STOR}$$

Changes in stored electricity ($\text{storage}^{\text{energy}}_{h,m,y,z,e}$) in two successive periods should be balanced by the charging ($\text{charge}_{h,m,y,z,e}$) and discharging ($\text{gen}_{h,m,y,z,e}$) processes:

$$\text{storage}^{\text{energy}}_{h,m,y,z,e} - \text{storage}^{\text{energy}}_{h-1,m,y,z,e} = \text{charge}_{h,m,y,z,e} - \text{gen}_{h,m,y,z,e}$$

In addition, the initial (when $h = h_{mstart}$) stored electricity ($\text{storage}^{\text{energy}}_{h=h_{\text{start}},m,y,z,e}$) of storage technology e in each month of each year can be calculated based on the proportion of the maximum storage capacity, as follows:

$$\text{storage}^{\text{energy}}_{h=h_{\text{start}},m,y,z,e} = \text{STORAGE}^{\text{energy}}_{m,y,z,e} \times \text{EP}_{e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall m, y, z, e \in \mathcal{STOR}$$

The instantaneous storage energy level ($\text{storage}^{\text{energy}}_{h,m,y,z,e}$) of storage technology e should not exceed the maximum energy storage capacity, as follows:

$$\text{storage}^{\text{energy}}_{h,m,y,z,e} \leq \text{EP}_{e} \times \text{cap}^{\text{existingtech}}_{y,z,e} \quad \forall h, m, y, z, e \in \mathcal{STOR}$$

class prepshot._model.storage.**AddStorageConstraints**(*model*)

Bases: object

Energy storage class.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

end_energy_storage_rule(*m*, *y*, *z*, *te*)

End energy storage.

Parameters

- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

energy_storage_balance_rule(*h*, *m*, *y*, *z*, *te*)

Energy storage balance.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

energy_storage_gen_rule(*h*, *m*, *y*, *z*, *te*)

Energy storage generation.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`energy_storage_up_bound_rule`(*h*, *m*, *y*, *z*, *te*)

Energy storage upper bound.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`init_energy_storage_rule`(*m*, *y*, *z*, *te*)

Initial energy storage.

Parameters

- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **te** (*str*) -- Technology.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

transmission

This module contains transmission related functions. We simplify the transmission of electricity as a transportation model. The model computes the transmission loss for each hour, in each time period, for each year, from z_{from} zone to z_{to} zone, as follows:

$$\text{import}_{h,m,y,z_{\text{from}},z_{\text{to}}} = \text{export}_{h,m,y,z_{\text{from}},z_{\text{to}}} \times \eta^{\text{trans}}_{z_{\text{from}},z_{\text{to}}} \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$

This model assumes that the transmitted power of each transmission line is only constrained by the transmission capacity between two zones as follows:

$$\text{import}_{h,m,y,z_{\text{from}},z_{\text{to}}} \leq \text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}} \times \Delta h \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$
$$\text{export}_{h,m,y,z_{\text{from}},z_{\text{to}}} \leq \text{cap}^{\text{existingline}}_{y,z_{\text{from}},z_{\text{to}}} \times \Delta h \quad \forall h, m, y, z_{\text{from}} \neq z_{\text{to}}$$

`class prepshot._model.transmission.`**AddTransmissionConstraints**(*model*)

Bases: `object`

Add constraints for transmission lines while considering multiple zones.

Parameters
model (*object*) --

__init__(*model*)

Initialize the class and add constraints.

Parameters
model (*object*) -- Model object depending on the solver.

Return type
None

trans_balance_rule(*h*, *m*, *y*, *z*, *z1*)

Transmission balance, i.e., the electricity imported from zone z1 to zone z should be equal to the electricity exported from zone z to zone z1 multiplied by the transmission line efficiency.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

trans_capacity_rule(*y*, *z*, *z1*)

Transmission capacity equal to the sum of the existing capacity and the new capacity in previous planned years.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
The expression of the model.

Return type
poi.ExprBuilder

trans_physical_rule(*y*, *z*, *z1*)

Physical transmission lines.

Parameters

- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

`trans_up_bound_rule`(*h*, *m*, *y*, *z*, *z1*)

Transmitted power is less than or equal to the transmission line capacity.

Parameters

- **h** (*int*) -- Hour.
- **m** (*int*) -- Month.
- **y** (*int*) -- Year.
- **z** (*str*) -- Zone.
- **z1** (*str*) -- Zone.

Returns
The constraint of the model.

Return type
poi.ConstraintIndex

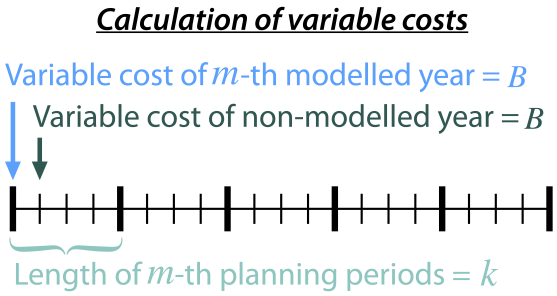
4.8.7 utils

This module contains utility functions for the model.

Here, we determine factors to covert future value to present value for costs and benefits. To account for the variable factor, fixed factor, and capital factor, we need to convert all future costs to their net present value. This means adjusting for the time value of money so that all costs are expressed in terms of today's dollars.

We also assume that variable cost and fixed cost for non-modelled years are assumed to be equal to the cost of the last modelled year preceding them. This allows for consistent comparison across different time periods and technologies.

Variable Factor



Given the following:

- Variable cost of modeled year: B
- Discount rate: r
- m-th modeled year: $m = y - y_{\text{min}}$
- Depreciation periods: n

The total present value can be calculated as follows:

$$\text{total present value} = \frac{B}{(1+r)^m} + \frac{B}{(1+r)^{m+1}} + \cdots + \frac{B}{(1+r)^{(m+k-1)}} \tag{4.1}$$

$$= B(1+r)^{(1-m)} \frac{1-(1+r)^k}{r} \tag{4.2}$$

And we can calculate the variable factor as follows:

$$\text{factor}_y^{var} = (1+r)^{1-m_y} \frac{1-(1+r)^{k_y}}{r} \tag{4.3}$$

$$m_y = y - y_{\text{min}} \tag{4.4}$$

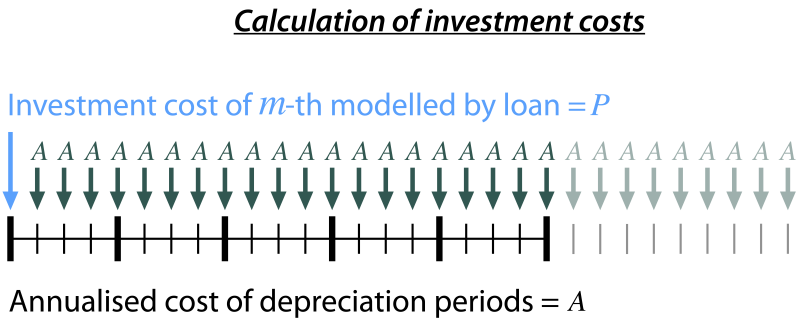
$$k_y = y_{\text{periods}} \tag{4.5}$$

Fixed Factor

We can equate the fixed factor with the variable factor as follows:

$$\text{factor}_y^{\text{fix}} = factor_y^{\text{var}}$$

Investment Factor



Given the following:

- Weighted Average Cost of Capital (WACC, or otherwise known as the interest rate): i
- Discount rate: r
- m-th modeled year: $m = y - y_{\text{min}}$
- Length of m-th planning periods: k

The total present value can be calculated as follows:

$$\text{total present value} = \frac{P}{(1+r)^m} \tag{4.6}$$

$$= \frac{\frac{A}{(1+i)} + \frac{A}{(1+i)^2} + \cdots + \frac{A}{(1+i)^n}}{(1+r)^m} \tag{4.7}$$

$$= A\frac{1-(1+i)^{-n}}{i} \times \frac{1}{(1+r)^m} \tag{4.8}$$

From the above, we can solve for the annualized cost of depreciation periods, A, as:

$$A = P\frac{i}{1-(1+i)^{-n}}$$

The capital recovery factor is then calculated as:

$$\text{capital recovery factor} = \frac{i}{1-(1+i)^{-n}}$$

Let's focus on the time periods that fall within the modelled time horizon (indicated in black colour). We can calculate the length of time periods, k, as follows:

$$k = y_{max} - y$$

Using k, we can calculate the net present value as follows:

$$\text{net present value} = \begin{cases} \frac{\frac{A}{(1+r)} + \frac{A}{(1+r)^2} + \cdots + \frac{A}{(1+r)^{min(n,k)}}}{(1+r)^m} \text{if} \quad n \leq k \\ \text{total present value} \quad \text{if } n > k \\ \frac{A\frac{1-(1+r)^{-k}}{r}}{(1+r)^m} = P\frac{i}{1-(1+i)^{-n}} \times \frac{1-(1+r)^{-k}}{r(1+r)^m} \quad \text{otherwise} \end{cases}$$

And we can calculate the investment factor as follows:

$$factor_y^{inv} = \frac{i}{1-(1+i)^{-n}} \times \frac{1-(1+r)^{-min(n,k)}}{r(1+r)^m}$$

prepshot.utils.**calc_cost_factor**(*discount_rate*, *modeled_year*, *year_min*, *next_modeled_year*)

Compute the variable and fixed cost factor while considering the multi-stage planning horizon.

Parameters

- **discount_rate** (*float*) -- The discount rate to apply.
- **modeled_year** (*int*) -- The year in which the cost occurs.
- **year_min** (*int*) -- The first year of the planning horizon. All costs are discounted to this year.
- **next_modeled_year** (*int*) -- The subsequent modeled year. The cost incurred bewteen modeled_year and modeled_year and next_modeled_year is calculated.

Returns
The computed cost factor.

Return type
float

Raises
ValueError -- if next_modeled_year < modeled_year.

Examples

Given annual cost incurred in 2025, next_modeled_year = 2030, and starting year = 2020, compute present value in 2020 of the cost incurred in 2025-2029:

```
>>> calc_cost_factor(0.05, 2025, 2020, 2030)
3.561871
```

prepshot.utils.**calc_inv_cost_factor**(*dep_period*, *interest_rate*, *year_built*, *discount_rate*, *year_min*, *year_max*)

Compute the investment cost factor. When the depreciation period is greater than the planning horizon, the investment cost factor is calculated by only considering the period within the planning horizon.

Parameters

- **dep_period** (*int*) -- Depreciation period, in years, i.e., lifetime of the infrastructure.

- **interest_rate** (*float*) -- Interest rate.
- **year_built** (*int*) -- Year of investment.
- **discount_rate** (*float*) -- Discount rate.
- **year_min** (*int*) -- Minimum year, i.e., the first year of the planning horizon.
- **year_max** (*int*) -- Maximum year, i.e., the last year of the planning horizon.

Returns
Investment cost factor.

Return type
float

Raises
ValueError -- If year_max <= year_min, year_max < year_built, or year_built < year_min.

Examples

Given a depreciation period of 20 years, interest rate of 0.05, year of investment in 2025, discount rate of 0.05, planning horizon from 2020 to 2050, compute the investment cost factor:

```
>>> calc_inv_cost_factor(20, 0.05, 2025, 0.05, 2020, 2050)
0.783526
```

If the depreciation perios is 100 years, compute the investment cost factor for the same scenario:

```
>>> calc_inv_cost_factor(100, 0.05, 2025, 0.05, 2020, 2050)
0.567482
```

prepshot.utils.**cartesian_product**(**args*)

Generate cartesian product of input iterables.

Parameters
args (*List[Union[int, str]]*) -- Iterables to be combined.

Returns
List of tuples representing the Cartesian product.

Return type
List[Tuple[Union[int, str]]]

Examples

Combine two lists [1, 2] and [7, 8]:

```
>>> cartesian_product([1, 2], [7, 8])
[(1, 7), (1, 8), (2, 7), (2, 8)]
```

prepshot.utils.**check_positive**(**values*)

Ensure all values are greater than 0.

Parameters
values (*Union[int, float]*) -- Values to be checked.

Raises
ValueError -- If any value is less than or equal to 0.

Return type
None

prepshot.utils.**interpolate_z_by_q_or_s**(*name*, *qs*, *zqv*)

Interpolate forebay water level (Z) by reservoir storage (S) or tailrace water level (Z) by the reservoir outflow (Q).

Parameters

- **name** (*str*) -- Code of the hydropower station.
- **qs** (*Union[np.ndarray, float]*) -- Reservoir storage or outflow values.
- **zqv** (*pandas.DataFrame*) -- DataFrame of ZQ or ZV values.

Returns
Interpolated values.

Return type
Union[np.ndarray, float]

4.8.8 output_data

This module contains functions to save solving results of models.

prepshot.output_data.**create_data_array**(*data*, *dims*, *unit*, *model*)

Create a xarray DataArray with specified data, dimensions, coordinates and units.

Parameters

- **data** (*dict*) -- The data to be included in the DataArray.
- **dims** (*list*) -- The list of dimentions of the data.
- **unit** (*str*) -- The unit of the data.
- **model** (*object*) -- The model object.

Returns
A DataArray with the specified data, dimensions, coordinates and units.

Return type
xr.DataArray

prepshot.output_data.**extract_results_hydro**(*model*)

Extracts results for hydro models.

Parameters
model (*object*) -- Model solved already.

Returns
A Dataset containing DataArrays for each attribute of the model.

Return type
xr.Dataset

prepshot.output_data.**extract_results_non_hydro**(*model*)

Extracts results for non-hydro models.

Parameters
model (*object*) -- Model object sloved already.

Returns
A Dataset containing DataArrays for each attribute of the model.

Return type
xr.Dataset

prepshot.output_data.**save_result**(*model*)

Extracts results from the provided model.

Parameters
model (*object*) -- The model object to extract results from and save.

Return type
None

prepshot.output_data.**save_to_excel**(*ds*, *output_filename*)

Save the results to an Excel file.

Parameters

- **ds** (*xr.Dataset*) -- Dataset containing the results.
- **output_filename** (*str*) -- The name of the output file.

Return type
None

prepshot.output_data.**update_output_filename**(*output_filename*, *args*)

Update the output filename based on the arguments.

Parameters

- **output_filename** (*str*) -- The name of the output file.
- **args** (*argparse.Namespace*) -- Arguments parsed by argparse.

Returns
The updated output filename.

Return type
str

4.9 Citation Guide

If you use PREP-SHOT for your work, please cite the appropriate publications. A list of all PREP-SHOT code related articles is available on GitHub archive and can be downloaded as single Bibtex file: `prepshot_publications.bib`.

4.9.1 Publications by Topic

If you use specific version of PREP-SHOT, please cite the appropriate publications presenting these versions of PREP-SHOT according to the following table:

Topic & Version used	Region	Publication to cite
Any	China Southern Power Grid	Liu, Z., & He, X. (2023). Balancing-oriented hydropower operation makes the clean energy transition more affordable and simultaneously boosts water security. Nature Water, 1(9), 778-789., https://www.nature.com/articles/s44221-023-00126-0
Hydropower-Wind-Solar Integration Capacity Configuration	Yalong River Basin	Cao, Y., Xu, B., Zhang, C., Li, F. F., & Liu, Z. (2025). Strategic Site-Level Planning of VRE Integration in Hydro-Wind-Solar Systems Under Uncertainty. Energy, 136523., https://doi.org/10.1016/j.energy.2025.136523
Integrated water-sediment-energy modeling framework	Lower Mekong Region	Xu, B., Liu, Z., Yan, S., Schmitt, RJP & He, X. (2025). Strategizing renewable energy transitions to preserve sediment transport integrity. Nature Sustainability, https://doi.org/10.1038/s41893-025-01626-5
Climate extremes and energy system resilience	Lower Mekong Region	Xie, J., Liu, Z., Yan, S. et al. Variable renewable energy pathways in the Lower Mekong Basin under projected river flow extremes. Commun Earth Environ 6, 928 (2025). https://doi.org/10.1038/s43247-025-02861-6

4.10 References

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Zhanwei Liu and Xiaogang He. Balancing-oriented hydropower operation makes the clean energy transition more affordable and simultaneously boosts water security. *Nature Water*, 1:778–789, 2023. doi:10.1038/s44221-023-00126-0.

PYTHON MODULE INDEX

p

INDEX

Symbols

A

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W