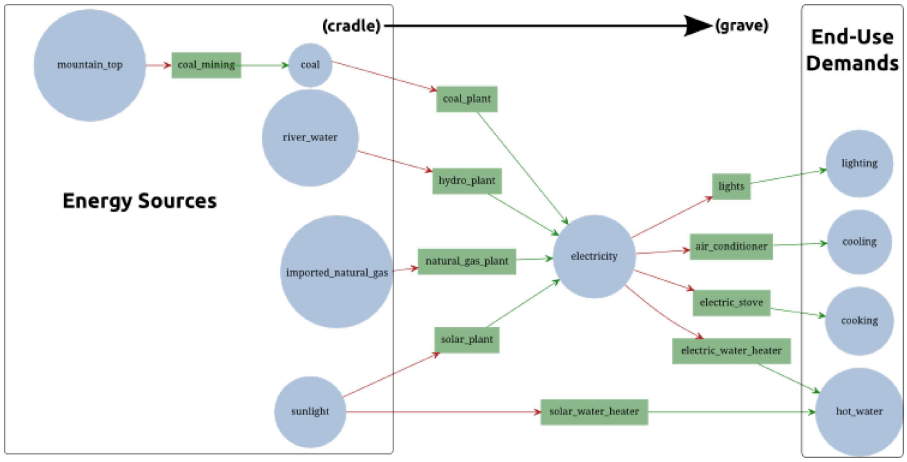
Mathematical Formulation

Contents

- Temoa Notation
- Treatment of Time
- Sets
- Parameters
- Decision Variables
- Equations
- General Caveats

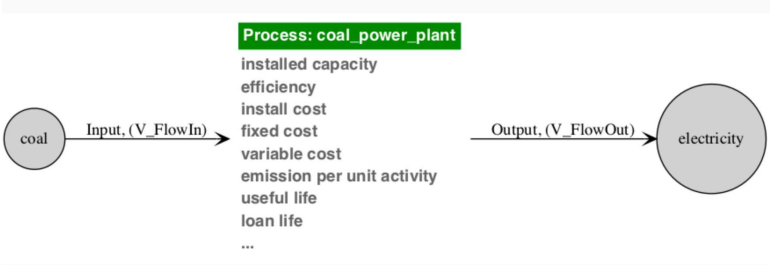
To understand this section, the reader will need at least a cursory understanding of mathematical optimization. We omit here that introduction, and instead refer the reader to `various`_ `available`_ `online`_ `sources`_. Temoa is formulated as an algebraic model that requires information organized into sets, parameters, variables, and equation definitions.

The heart of Temoa is a technology explicit energy system optimization model. It is an algebraic network of linked processes – where each process is defined by a set of engineering characteristics (e.g. capital cost, efficiency, capacity factor, emission rates) – that transform raw energy sources into end-use demands. The model objective function minimizes the present-value cost of energy supply by optimizing installed capacity and its utilization over time.



A common visualization of energy system models is a directed network graph, with energy sources on the left and end-use demands on the right. The modeler must specify the end-use demands to be met, the technologies defined within the system (rectangles), and the inputs and outputs of each (red and green arrows). The circles represent distinct energy carriers that connect technologies within the energy system network.

The most fundamental tenet of the model is the understanding of energy flow, treating all processes as black boxes that take inputs and produce outputs. Specifically, Temoa does not care about the inner workings of a process, only its global input and output characteristics. In this vein, the above graphic can be broken down into process-specific elements. For example, the coal power plant takes as input coal and produces electricity, and is subject to various costs (e.g. variable costs) and constraints (e.g. efficiency) along the way.



The modeler defines the processes and engineering characteristics through a combination of sets and parameters, described in the next few sections. Temoa then utilizes these parameters, along with the associated technology-specific decision variables for capacity and activity, to create the objective function and constraints that are used during the optimization process.

Temoa Notation

In the mathematical notation, we use CAPITALIZATION to denote a container, like a set, indexed variable, or indexed parameter. Sets use only a single letter, so we use the lower case to represent an item from the set. For example, T represents the set of all technologies and t represents a single item from T.

- Variables are named V_VarName within the code to aid readability. However, in the documentation where there is benefit of italics and other font manipulations, we elide the 'V_' prefix.
- In all equations, we **bold** variables to distinguish them from parameters. Take, for example, this excerpt from the Temoa default objective function:

$$C_{variable} = \sum_{r,p,s,d,i,t,v,o \in \Theta_{VC}} (VC_{r,p,t,v} \cdot R_p \cdot \mathbf{FO}_{r,p,s,d,i,t,v,o})$$

Note that $C_{variable}$ is not bold, as it is a temporary variable used for clarity while constructing the objective function. It is not a structural variable and the solver never sees it.

- Where appropriate, we put the variable on the right side of the coefficient. In other words, this is not a preferred form of the previous equation:

$$C_{variable} = \sum_{r,p,s,d,i,t,v,o \in \Theta_{VC}} (\mathbf{FO}_{r,p,s,d,i,t,v,o} \cdot VC_{r,p,t,v} \cdot R_p)$$

- We generally put the limiting or defining aspect of an equation on the right hand side of the relational operator, and the aspect being limited or defined on the left hand side. For example, equation (1) defines Temoa's mathematical understanding of a process capacity ($\mathbf{CAP}$) in terms of that process' activity ($\mathbf{ACT}$):

$$(\mathrm{CFP}_{r,s,d,t,v} \cdot \mathrm{C2A}_{r,t} \cdot \mathrm{SEG}_{s,d}) \cdot \mathbf{CAP}_{r,t,v} = \sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} + \sum_{I,O} \mathbf{CUR}_{r,p,s,d,i,t,v,o}$$

$$\forall \{r,p,s,d,t,v\} \in \Theta_{\mathrm{FO}}$$

- We use the word 'slice' to refer to the tuple of season and time of day $\{s,d\}$. Note that these time slices are user-defined, and can represent time ranging large blocks of time (e.g., winter-night) to every hour in a given season.
- We use the word 'process' to refer to the tuple of technology and vintage ($\{t,v\}$). For example, solar PV (technology) installed in 2030 (vintage).

Treatment of Time

Temoa's conceptual model of *time* is broken up into three levels, and energy supply and demand is balanced at each of these levels:

- **Periods** - consecutive blocks of years, marked by the first year in the period. For example, a two-period model might consist of $\mathrm{P}^f = \{2010, 2015, 2025\}$, representing the two periods of years from 2010 to 2015, and from 2015 to 2025. It is up to the model builder whether this represents start of year to start of year or end of year to end of year. Note that the last period element (2025) does not represent a new time period, but rather defines the end of the second time period and therefore the planning horizon.
- **Seasonal** - Each year is divided into one or more seasons. What a "season" represents depends on the `time_sequencing` mode chosen in the configuration file (see below). Each season carries a `segment_fraction` value in the `time_season` table specifying the fraction of the year it represents.
- **Daily** - Within each season, the day is subdivided into one or more time-of-day segments. Each segment has an associated `hours` value in the `time_of_day` table indicating how many

hours it represents (e.g. 8 hours for “Day”, 16 hours for “Night”). Less detailed databases may use a single segment covering the full day.

We use the word ‘slice’ or ‘timeslice’ to refer to the tuple of season and time of day $\{s, d\}$. The fraction of a year represented by each slice is computed automatically:

$$SEG_{s,d} = \text{segment\_fraction\_per\_season}(s) \times \frac{\text{hours}(d)}{\sum_{d'} \text{hours}(d')}$$

The sum of $SEG_{s,d}$ over all (s, d) pairs must equal 1.

Time Sequencing Modes

Temoa v4 supports four modes for interpreting the meaning and ordering of seasons, controlled by the `time_sequencing` setting in the configuration TOML file:

- **consecutive_days** — Each season represents a single day, and the seasons are treated as consecutive days in order. Use this when modeling e.g., a representative week (7 seasons) or a full year (365 seasons). Inter-season state (e.g. for storage) carries forward naturally from one season to the next, so the `time_season_sequential` table can be left empty (more on that below).
- **seasonal_timeslices** *(default)* — Each season represents a sequential slice of the year containing one or many days (e.g. Winter, Spring, Summer, Fall). The true chronological sequence is assumed to follow the `time_season` ordering, so the `time_season_sequential` table can be left empty. This is the traditional Temoa time representation.
- **representative_periods** — Each season represents a block of days that may not be contiguous in time (e.g. a “typical summer weekday” drawn from several months). If the model uses inter-season constraints such as seasonal storage or inter-season ramping, the true chronological sequence must be defined in the `time_season_sequential` table. Technologies using seasonal storage must also be flagged in the `technology` table.
- **manual** — The sequence of time slices is defined explicitly by the modeler in an optional `time_manual` table. This is an advanced feature provided in case none of the above modes are suitable, and is not recommended for most users.

The selected mode, in combination with the sequence of the `time_season` and `time_of_day` tables, is used to construct the `time_next` dictionary, which maps each (s, d) time slice to its successor in the chronological sequence, (s_{next}, d_{next}).

The `days_per_period` configuration setting (default: 365) specifies how many days each planning period’s representative year encompasses. This is used to scale flow variables correctly. For example, reduce it to 7 when modeling a single representative week.

Time Season Sequential

When using the **representative_periods** time sequencing mode, the seasons in `time_season` may represent non-contiguous blocks of time (e.g. a "typical summer weekday" drawn from several calendar months). For constraints that depend on inter-season ordering — seasonal storage and inter-season ramping — the model needs to know the true chronological sequence showing how the representative seasons are stitched together to form a complete year.

The `time_season_sequential` table provides this mapping. Each row defines a *sequential season* (`seas_seq`) that references one of the `time_season` entries and carries its own `segment_fraction` and an integer `sequence` column that determines the chronological order. Because the same representative season may appear more than once in the reconstructed year (e.g. "typical summer weekday" could appear for June, July, and August), the sequential table can contain multiple entries that map back to the same `time_season` row, each with its own fraction.

From this table Temoa builds two internal dictionaries:

- `time_next_sequential` — maps each sequential season to its successor, forming a circular chain that represents the annual cycle.
- `sequential_to_season` — maps each sequential season back to its parent representative season in `time_season`.

These dictionaries are consumed by:

- The **seasonal storage energy constraint**, which chains the storage state of charge across sequential seasons in chronological order.
- The **ramp up/down season constraints**, which limit the rate of activity change at season boundaries that are adjacent in the sequential ordering but not necessarily adjacent in the `time_season` set.

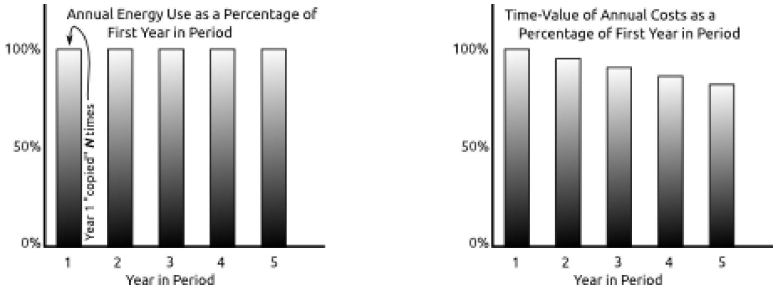
For the **consecutive_days** and **seasonal_timeslices** modes, the `time_season_sequential` table may be left empty — the model derives the chronological order directly from the `time_season` set.

Periods

There are two specifiable period sets: `time_exist` (P^e) and `time_future` (P^f). The `time_exist` set contains periods before `time_future`. Its primary purpose is to specify the vintages for capacity that exists prior to the model optimization. The `time_future` set contains the future periods that the model will optimize. As this set must contain only integers, Temoa interprets the elements to be the boundaries of each period of interest. Thus, this is an ordered set and Temoa uses its elements to automatically calculate the length of each optimization period; modelers may exploit this to create variable period lengths within a given input database. Temoa "names" each optimization period by the first year, and makes them easily accessible via the `time_optimize` set. This final "period" set is not user-specifiable, but is an exact duplicate of `time_future`, less the largest element. In the above example, since $\mathrm{P}^f = \{2010, 2015, 2025\}$, `time_optimize` does not contain 2025: $\mathrm{P}^o = \{2010, 2015\}$.

Temoa assumes that all elements of the `time_exist` and `time_future` sets are integers. Further, these sets are assumed to be ordered, such that the minimum element is "naught". For example, if $\mathrm{P}^f = \{2015, 2020, 2030\}$, then $P_0 = 2015$. In other words, the capital P with the naught subscript indicates the first element in the `time_future` set.

One final note on periods: rather than optimizing each year within a period individually, Temoa makes the simplifying assumption that each time period contains n copies of a single, representative year. Temoa optimizes capacity and activity for just this characteristic year within each time period, assuming the results for different years in the same time period are identical. The Temoa objective function, however, accounts for the total cost across all years in all model time periods. Figure 3.3 gives a graphical explanation of the annual delineation.



The left graph is of energy, while the right graph is of the annual costs. The energy used in a period by a process is the same for all years (with exception for those processes that cease their useful life mid-period). However, even though the costs incurred will be the same, the time-value of money changes due to the discount-rate. As the fixed costs of a process are tied to the length of its useful life, those processes that do not fall on a period boundary require unique time-value multipliers in the objective function.

As noted above, Temoa allows the modeler to subdivide each year into a set of time slices, comprised of a season and a time of day. Unlike `time_future`, there is no restriction on what labels the modeler may assign to the `time_season` and `time_of_day` set elements. Each season carries a `segment_fraction` (fraction of the year), and each time-of-day segment carries an `hours` value. These are combined to compute the `segment_fraction` for each (s, d) slice as described above.

Sets

In a mathematical model like Temoa, sets are used to index parameters and variables. To enable the modeler to translate between the algebraic formulation, input database, and model code, we provide the mathematical notation used in the model formulation, as well as the associated names in the Python code and database schema in the tables below. The code representation is more verbose than the algebraic version, using full words.

Our discussion of sets is broken down into several logical categories to make it easier to parse. The **first group of sets pertains to Temoa's treatment of time**, as shown below. **Note:** Some entries in the "Database Table" column are comma-separated. In those cases, the first element refers to the name of the database table, and the second refers to specific column within the database table used to identify a specific subset. For example, in the first table below, `time_period` is the master set and `time_exist` is a subset that defines the user-defined model time periods to define historical technology vintages that exist prior to the model's time horizon for optimization.

Set	Database Table	Model Element	Notes
P^e	`time_period, flag = e`	`time_exist`	model periods prior to the beginning of the optimization time horizon
P^f	`time_period, flag = f`	`time_future`	model periods within the optimization time horizon
$^*\mathrm{P}^o$	`time_period`	`time_optimize`	model time periods to optimize, $(\mathrm{P}^f - \max(\mathrm{P}^f))$; the last time period is removed as it represents the end of the final period
$^*\mathrm{V}$	`time_period`	`vintage_exist`, `vintage_optimize`, `vintage_all`	tech vintages, $(\mathrm{P}^e \cup \mathrm{P}^o)$, derived from time period set and used to track technology vintages across time periods
D	`time_of_day`	`time_of_day`	intraday time divisions; each entry carries an `hours` value (default 1) specifying how many hours it represents
S	`time_season`	`time_season`	intra-annual divisions (e.g. seasons or representative days); each entry carries a `segment_fraction` giving its share of the year; ordered by `sequence` column
	`time_season_sequential`	`time_season_sequential`, `ordered_season_sequential`	superimposed sequential seasons used for seasonal storage and inter-season ramping; only required when `time_sequencing = 'representative_periods'`

The sets in the table below define Temoa's representation of **regions**.

Set	Database Table	Model Element	Notes
R	`region`	`regions`	distinct geographical regions
	`region`	`regional_indices`	set of all the possible combinations of interregional exchanges plus original region indices
		`regional_global_indices`	set of all used combinations of interregional exchanges plus original region indices and groups

The sets below define how technologies are represented within Temoa. Because technologies can serve many different functions across an energy system, we need to define a large number of **technology subsets**.

Set	Database Table	Model Element	Notes
$^*\mathrm{T}$	`technology`	`tech_all`	all technologies to be modeled (in v4, all technologies are production-type: $T = T^p$)
T^p	`technology, flag LIKE 'p%'`	`tech_production`	all production technologies, including baseload and storage ($T^b \cup T^s \subset T^p$)
T^b	`technology, flag = pb`	`tech_baseload`	baseload electric generators, which have constant output across intraday time segments ($T^b \subset T$)
T^s	`technology, flag = ps`	`tech_storage`	all storage technologies ($T^s \subset T$)
T^a	`technology, annual = 1`	`tech_annual`	technologies that produce constant annual output ($T^a \subset T$)
T^c	`technology, curtail = 1`	`tech_curtailment`	technologies with curtailable output and no upstream cost ($T^c \subset T$)
T^f	`technology, flex = 1`	`tech_flex`	technologies producing excess commodity flows ($T^f \subset T$)
T^x	`technology, exchange = 1`	`tech_exchange`	technologies used for interregional commodity flows ($T^x \subset T$)
T^{ur}	`ramp_up_hourly`	`tech_upramping`	electric generators with a ramp up hourly rate limit; derived from `ramp_up_hourly` table ($T^{ur} \subset T$)
T^{dr}	`ramp_down_hourly`	`tech_downramping`	electric generators with a ramp down hourly rate limit; derived from `ramp_down_hourly` table ($T^{dr} \subset T$)
T^{ret}	`technology, retire = 1`	`tech_retirement`	technologies allowed to retire before end of life ($T^{ret} \subset (T - T^u)$)
T^u	`technology, unlim_cap = 1`	`tech_uncap`	technologies that have no bound on capacity ($T^u \subset T$)
T^{ss}	`technology, flag = 'ps' AND seas_stor = 1`	`tech_seasonal_storage`	seasonal storage technologies; requires both storage flag and seas_stor column ($T^{ss} \subset T^s$)

Set	Database Table	Model Element	Notes
	`tech_group`	`tech_group_names`	named groups for use in group constraints
	`tech_group_member`	`tech_group_members`	each technology belonging to each group
T^e	`existing_capacity`	`tech_exist`	existing technologies with a past vintage ($T^e \subset T$)

The sets below define **commodities** that are consumed and produced by different energy technologies.

Set	Database Table	Model Element	Notes
C^d	`commodity, flag = d`	`commodity_demand`	end-use demand commodities, representing the final consumer demands
C^e	`commodity, flag = e`	`commodity_emissions`	emission commodities (e.g., CO_2 NO_x); filtered by flag
C^p	`commodity, flag IN (p, wp, s, a, wa)`	`commodity_physical`	superset of physical, source, annual, and their waste variants; includes all non-demand, non-emission commodities
C^w	`commodity, flag IN (w, wa , wp)`	`commodity_waste`	commodity whose production can be greater than its consumption; can be physical, annual, or neither (not balanced, all wasted)
C^a	`commodity, flag IN (a, wa)`	`commodity_annual`	commodities whose flows are only balanced over each period, not per-timeslice ($\mathrm{C}^a \subset \mathrm{C}^p$)
${}^*\mathrm{C}^l$		`commodity_flex`	derived set of commodities produced by a flex technology ($\mathrm{C}^l \subset \mathrm{C}^p$); auto-populated from `tech_flex` outputs
C^s	`commodity, flag = s`	`commodity_source`	primary source commodities, not balanced by `CommodityBalance_constraint`
${}^*\mathrm{C}^c$		`commodity_carrier`	union of physical, demand, and waste commodities, ($\mathrm{C}_p \cup \mathrm{C}_d \cup \mathrm{C}^w$)
${}^*\mathrm{C}$		`commodity_all`	union of all commodity sets; union of carrier and emissions commodities

There is an additional set that defines **operators (=, <, >)**. While not strictly necessary, defining these operators as a set allows modelers to express constraints more efficiently.

Set	Database Table	Model Element	Notes
	`operator`	`operator`	constraint operators

As indicated below, there are additional sets that are derived within the model code and thus do not appear in the database schema.

Set	Model Element	Notes
	`tech_with_capacity`	technologies eligible for capacitization; computed as tech_all - tech_uncap
	`tech_or_group`	technologies or groups combined; union of tech_group_names | tech_all
${}^*\mathrm{C}^c$	`commodity_carrier`	physical energy carriers and end-use demands; union of physical, demand, and waste commodities
${}^*\mathrm{C}$	`commodity_all`	union of all commodity sets; union of carrier and emissions commodities
T^e	`tech_exist`	technologies with existing capacity; derived from existing_capacity table

There are also python dictionaries and sets used to specify internal data structures during model construction and are not considered formal model elements:

- `process_inputs`, `process_outputs`, `process_loans`
- `active_flow_rpsditvo`, `active_flow_rpitvo`
- Various vintage and operational tracking dictionaries
- Time sequencing dictionaries (`time_next`, `time_next_sequential`)

Parameters

A summary table of input parameters is provided below, followed by a more detailed description of each.

Parameters are indexed by the sets defined above and used to specify input data. In the leftmost column below, the subscripts indicate the sets used to index the parameter. As with sets, we categorize parameters thematically and present them below in separate tables.

Below are the **time-related** parameters.

Parameter	Database Table	Model Element	Notes
SFS_s	`time_season`	`segment_fraction_per_season`	per-season year f from the `segment` column of `time_`
	`time_of_day`	`time_of_day_hours`	hours per time-o loaded from the of `time_of_day` (
$\mathrm{SEG}_{s,d}$	*(computed)*	`segment_fraction`	fraction of year re each (s, d) tuple; `segment_fractio` `× hours(d) / Σ h`
	(config)	`time_sequencing`	time sequencing `consecutive_day` `seasonal_timesl` `representative_` `manual`
	(config)	`days_per_period`	number of days p (default: 365); set key in the configu file, not in the da
$\mathrm{SFS}_{s^{seq}}$	`time_season_sequential`	`segment_fraction_per_sequential_season`	per-sequential-se fraction; loaded f `segment_fractio` `time_season_seq`

Parameters in the table below relate to **capacity and its performance characteristics**.

Parameter	Database Table	Model Element	Notes
$\mathrm{C2A}_{r,t}$	`capacity_to_activity`	`capacity_to_activity`	converts from capacity to activity units
$\mathrm{PRC}_{r,t}$	`planning_reserve_credit`	`planning_reserve_credit`	fraction of installed capacity that can be relied upon (default 0)
$\mathrm{CFT}_{r,s,d,t}$	`capacity_factor_tech`	`capacity_factor_tech`	technology-specific capacity factor
$\mathrm{CFP}_{r,s,d,t,v}$	`capacity_factor_process`	`capacity_factor_process`	process-specific capacity factor; allows capacity factor to change with technology vintage
$\mathrm{ECAP}_{r,t,v}$	`existing_capacity`	`existing_capacity`	installed capacity that exists prior to first model time period
PRM_{r_g,t_g}	`planning_reserve_margin`	`planning_reserve_margin`	required excess of credited capacity above demand in each time slice, as a fraction of demand
ORM_{r_g,t_g}	`operating_reserve_margin`	`operating_reserve_margin`	required excess of available (derated) output above demand in each time slice, as a fraction of that demand
$\mathrm{ORD}_{r,s,t}$	`operating_reserve_derate`	`operating_reserve_derate`	fraction of available output that can be relied upon in a given season (default 1)
$\mathrm{RUH}_{r,t}$	`ramp_up_hourly`	`ramp_up_hourly`	hourly rate at which generation techs can ramp output up
$\mathrm{RDH}_{r,t}$	`ramp_down_hourly`	`ramp_down_hourly`	hourly rate at which generation techs can ramp output down

Parameters in the table below relate to the specification of **costs**.

Parameter	Database Table	Model Element	Notes
$\mathrm{CF}_{r,p,t,v}$	`cost_fixed`	`cost_fixed`	fixed operations & maintenance cost
$\mathrm{CI}_{r,t,v}$	`cost_invest`	`cost_invest`	tech-specific investment cost
$\mathrm{CV}_{r,p,t,v}$	`cost_variable`	`cost_variable`	variable operations & maintenance (O&M) cost
$\mathrm{CE}_{r,p,e}$	`cost_emission`	`cost_emission`	emission costs
GDR	`metadata_real`	`global_discount_rate`	global rate used to convert future time period costs to the present cost
DLR	`metadata_real`	`default_loan_rate`	default loan rate used to amortize investment costs
$\mathrm{LLP}_{r,t,v}$	`loan_lifetime_process`	`loan_lifetime_process`	process-specific loan term (default=lifetime_process)
$\mathrm{LR}_{r,t,v}$	`loan_rate`	`loan_rate`	process-specific interest rate on investment cost

Parameters in the table below relate to the specification of **technology efficiency and performance**.

Parameter	Database Table	Model Element	Notes
$\mathrm{EFF}_{r,i,t,v,o}$	`efficiency`	`efficiency`	technology- and commodity-specific conversion efficiency
$\mathrm{EFFV}_{r,s,d,i,t,v,o}$	`efficiency_variable`	`efficiency_variable`	optional time slice multiplier on efficiency (no period index; applies to all periods)
$\mathrm{LTT}_{r,t}$	`lifetime_tech`	`lifetime_tech`	technology-specific lifetime (default=40 years)
$\mathrm{LTP}_{r,t,v}$	`lifetime_process`	`lifetime_process`	tech- and vintage-specific lifetime (default=lifetime_tech)
$\mathrm{LSC}_{r,p,t,v}$	`lifetime_survival_curve`	`lifetime_survival_curve`	surviving fraction of original capacity
$\mathrm{SD}_{r,t}$	`storage_duration`	`storage_duration`	storage duration per technology, specified in hours

Parameters in the table below relate to the specification of **end-use demands**.

Parameter	Database Table	Model Element	Notes
$\mathrm{DEM}_{r,p,c}$	`demand`	`demand`	end-use demand by region-period-commodity
$\mathrm{DSD}_{r,s,d,c}$	`demand_specific_distribution`	`demand_specific_distribution`	fractional annual demand by time slice (s,d)

Parameters in the table below relate to the specification of **emissions**.

Parameter	Database Table	Model Element	Notes
$\mathrm{EAC}_{r,e,i,t,v,o}$	`emission_activity`	`emission_activity`	activity-based emissions rate
$\mathrm{EE}_{r,t,v,e}$	`emission_embodied`	`emission_embodied`	emissions associated with the creation of capacity; embodied emissions
$\mathrm{EEOL}_{r,t,v,e}$	`emission_end_of_life`	`emission_end_of_life`	emissions associated with the retirement/end of life of capacity

Parameters in the table below relate to the specification of **absolute limits on capacity, activity and emissions**.

Parameter	Database Table	Model Element	Notes
$\mathrm{LC}_{r,p,t}$	`limit_capacity`	`limit_capacity`	limit on capacity by period; `tech_or_group` column accepts a technology name or group name
$\mathrm{LNC}_{r,t,v}$	`limit_new_capacity`	`limit_new_capacity`	limit on new capacity deployment by vintage; `tech_or_group` column accepts a technology name or group name
$\mathrm{LA}_{r,p,t}$	`limit_activity`	`limit_activity`	limit on activity by region and period; `tech_or_group` column accepts a technology name or group name
$\mathrm{LE}_{r,p,e}$	`limit_emission`	`limit_emission`	limit on emissions by region and period
$\mathrm{LS}_{r,t}$	`limit_resource`	`limit_resource`	cumulative activity limit across time periods (not supported in myopic); `tech_or_group` column accepts a technology name or group name

Parameters in the table below relate to the specification of **operational and split limits**.

Parameter	Database Table	Model Element	Notes
$\mathrm{LACF}_{r,t,v,o}$	`limit_annual_capacity_factor`	`limit_annual_capacity_factor`	annual capacity factor limits; indexed by vintage (not period); `tech_or_group` column accepts a technology name or group name
$\mathrm{LSCF}_{r,s,t}$	`limit_seasonal_capacity_factor`	`limit_seasonal_capacity_factor`	seasonal capacity factor limits (no period index; applies to all periods)
$\mathrm{LSF}_{r,s,d,t}$	`limit_storage_level_fraction`	`limit_storage_fraction`	limit on storage level in any time slice
$\mathrm{TIS}_{r,p,i,t}$	`limit_tech_input_split`	`limit_tech_input_split`	technology input split constraints specifying input fuel ratio at time slice level
$\mathrm{TISA}_{r,p,i,t}$	`limit_tech_input_split_annual`	`limit_tech_input_split_annual`	technology input split constraints specifying input fuel ratio at average annual level
$\mathrm{TOS}_{r,p,t,o}$	`limit_tech_output_split`	`limit_tech_output_split`	technology split constraints specifying output fuel

Parameter	Database Table	Model Element	Notes
			ratio at time slice level
$\mathrm{TOSA}_{r,p,t,o}$	`limit_tech_output_split_annual`	`limit_tech_output_split_annual`	technology split constraints specifying output fuel ratio at average annual level

Parameters in the table below relate to the specification of **share limits**.

Parameter	Database Table	Model Element	Notes
$\mathrm{LCS}_{r,p,g_1,g_2}$	`limit_capacity_share`	`limit_capacity_share`	capacity share limits between technology groups
$\mathrm{LAS}_{r,p,g_1,g_2}$	`limit_activity_share`	`limit_activity_share`	activity share limits between technology groups
$\mathrm{LNCS}_{r,g_1,g_2,v}$	`limit_new_capacity_share`	`limit_new_capacity_share`	new capacity share limits

Parameters in the table below relate to the specification of **policy**.

Parameter	Database Table	Model Element	Notes
$\mathrm{LIT}_{r,t,e,t'}$	`linked_tech`	`linked_techs`	dummy techs used to convert CO2 emissions to physical commodity

Parameters in the table below relate to the specification of **construction and end-of-life**.

Parameter	Database Table	Model Element	Notes
$\mathrm{CON}_{r,i,t,v}$	`construction_input`	`construction_input`	construction input requirements that represent commodities consumed by creation of process capacity
$\mathrm{EOLO}_{r,t,v,o}$	`end_of_life_output`	`end_of_life_output`	end-of-life outputs that represent commodities produced by retirement/end of life of capacity

Parameters in the table below are **computed in code (i.e., model-derived) and therefore do not appear in the database**.

Parameter	Database Table	Model Element	Notes
${}^*\mathrm{LEN}_p$		`period_length`	number of years in period p; computed from time periods
${}^*\mathrm{LA}_{t,v}$		`loan_annualize`	loan amortization by tech and vintage; based on DR_t; computed from loan rate and lifetime
${}^*\mathrm{PLF}_{r,p,t,v}$		`process_life_frac`	fraction of available process capacity by region and period; computed from process life fraction

The tables below specify **model outputs.** When constructing a new database, they are left blank. These tables are auto-populated by the model after successful run completion.

- `output_dual_variable`
- `output_objective`
- `output_curtailment`
- `output_net_capacity`
- `output_built_capacity`
- `output_retired_capacity`
- `output_flow_in`
- `output_flow_out`
- `output_flow_out_summary`
- `output_storage_level`
- `output_emission`
- `output_cost`

Finally, the database tables below do not have a directed mapping to the model code.

Database Table	Purpose	Notes
`myopic_efficiency`	Myopic mode efficiency	alternative efficiency for myopic optimization
`sector_label`	Sectoral classification	used in output tables only

efficiency

$EFF_{r \in R, i \in C_p, t \in T, v \in V, o \in C_c}$

We present the efficiency (EFF) parameter first as it is one of the most critical model parameters. Beyond defining the conversion efficiency of each process, Temoa also utilizes the indices to understand the valid input $\rightarrow$ process $\rightarrow$ output paths for energy. For instance, if a modeler does not specify an efficiency for a 2020 vintage coal power plant, then Temoa will recognize any mention of a 2020 vintage coal power plant elsewhere as an error. Generally, if a process is not specified in the efficiency table, [efficiency_table] Temoa assumes it is not a valid process and will provide the user a warning with pointed debugging information.

efficiency_variable

$EFF_{r \in R, s \in S, d \in D, i \in C_p, t \in T, v \in V, o \in C_c}$

When a technology's conversion efficiency varies by time of day or season — for example, a heat pump whose efficiency differs with temperature — the modeler can use `efficiency_variable` to specify these time-slice-dependent efficiency values. If not specified for a given process, it defaults to 1, meaning the base `efficiency` value applies uniformly. Note that there is no period index: the time-varying efficiency applies to all periods.

capacity_factor_tech

$CFT_{r \in R, s \in S, d \in D, t \in T}$

Temoa indexes the `capacity_factor_tech` parameter by season, time-of-day, and technology.

capacity_factor_process

$CFP_{r \in R, s \in S, d \in D, t \in T, v \in V}$

In addition to capacity_factor_tech, there may be cases where different vintages of the same technology have different capacity factors. For example, newer vintages of wind turbines may have higher capacity

factors. So, `capacity_factor_process` allows users to specify the capacity factor by season, time-of-day, technology, and vintage.

capacity_to_activity

$C2A_{r \in R, t \in T}$

Capacity and Activity are inherently two different units of measure. Capacity represents the maximum flow of energy per time ($\frac{energy}{time}$), while Activity is a measure of total energy actually produced. However, there are times when one needs to compare the two, and this parameter makes those comparisons more natural. For example, a capacity of 1 GW for one year works out to an activity of

$$1GW \cdot 8,760 \frac{hr}{yr} \cdot 3,600 \frac{sec}{hr} \cdot 10^{-6} \frac{P}{G} = 31.536 \frac{PJ}{yr}$$

or

$$1GW \cdot 8,760 \frac{hr}{yr} \cdot 10^{-3} \frac{T}{G} = 8.75TWh$$

When comparing one capacity to another, the comparison is easy, unit wise. However, when one *needs* to compare capacity and activity, how does one reconcile the units? One way to think about the utility of this parameter is in the context of the question: "How much activity would this capacity create, if used 100% of the time?"

cost_fixed

$CF_{r \in R, p \in P, t \in T, v \in V}$

The `cost_fixed` parameter specifies the fixed cost associated with any process. Fixed costs are those that must be paid, regardless of how much the process is utilized. For instance, if the model decides to build a nuclear power plant, even if it decides not utilize the plant, the model must pay the fixed costs. These costs are in addition to the capital cost, so once the capital is paid off, these costs are still incurred every year the process exists.

Temoa's default objective function assumes the modeler has specified this parameter in units of currency per unit capacity ($\frac{Dollars}{UnitCap}$).

cost_invest

$CI_{r \in R, t \in T, v \in P}$

The `cost_invest` parameter specifies the process-specific investment cost. Unlike the `cost_fixed` and `cost_variable` parameters, `cost_invest` only applies to vintages of technologies within the model

optimization horizon (P^o). Like `cost_fixed`, `cost_invest` is specified in units of cost per unit of capacity and is only used in the default objective function ($\frac{Dollars}{UnitCap}$).

cost_variable

$CV_{r \in R, p \in P, t \in T, v \in V}$

The `cost_variable` parameter represents the cost of a process-specific unit of activity. Thus the incurred variable costs are proportional to the activity of the process.

cost_emission

$CE_{r \in R, p \in P, e \in C^e}$

The `cost_emission` parameter specifies a cost per unit of emission for a given emissions commodity in each period. This allows the modeler to penalize emission production, for example via a carbon tax. The cost is applied to total emissions of commodity e in period p.

construction_input

$CON_{r \in R, i \in C^p, t \in T \setminus T^u, v \in V}$

The `construction_input` parameter allows the modeller to attach commodity input flows to the production of new capacity, in units of activity per unit capacity. Assumes that capacity is produced evenly over years in its vintage period.

demand

$DEM_{r \in R, p \in P, c \in C^d}$

The `demand` parameter allows the modeler to define the total end-use demand levels for all periods. In combination with the `efficiency` parameter, this parameter is the most important because without it, the rest of model has no incentive to build anything. This parameter specifies the end-use demands that appear at the far right edge of the system diagram.

To specify a non-uniform distribution of demand across time slices, use the `demand_specific_distribution` (DSD) parameter, described next.

demand_specific_distribution

$DSD_{r \in R, s \in S, d \in D, c \in C^d}$

If there is an end-use demand that varies over the course of a day or across seasons – for example, heating or cooling in the summer or winter – the modeler may specify the fraction of annual demand occurring in each time slice. The sum of DSD for each demand commodity c must be 1. If the modeler does not define DSD for a demand commodity, Temoa automatically populates it using the `segment_fraction` values, resulting in a uniform (flat) distribution proportional to the length of each time slice.

emission_activity

$$EAC_{e \in C_e, \{r,i,t,v,o\} \in \Theta_{\text{efficiency}}}$$

Temoa currently has two methods for enabling a process to produce an output: the `efficiency` parameter, and the `emission_activity` parameter. Where the `efficiency` parameter defines the amount of output energy a process produces per unit of input, the `emission_activity` parameter allows for secondary outputs. As the name suggests, this parameter was originally intended to account for emissions per unit activity, but it more accurately describes *parallel* activity. It is restricted to emissions accounting (by the $e \in C^e$ set restriction).

emission_embodied

$$EE_{r \in R, t \in T \setminus T^u, v \in V, e \in C_e}$$

Like the emission_activity parameter, but attaches emission outputs to the creation of capacity instead of activity flows. Assumes that capacity is produced evenly over each year in the deployment vintage.

emission_end_of_life

$$EEOL_{r \in R, t \in T \setminus T^u, v \in V, e \in C_e}$$

Like emission_embodied, but attaches emissions to the retirement/end of life of capacity rather than production of capacity. Assumes that retirement or end of life occur evenly over years in that period.

end_of_life_output

$$EOLO_{r \in R, t \in T \setminus T^u, v \in V, o \in C_p}$$

Like construction_input, but attaches flows to the retirement/end of life of capacity rather than production of capacity. Assumes that retirement or end of life occur evenly over years in that period.

existing_capacity

$$ECAP_{r \in R, t \in T, v \in \mathrm{P}^e}$$

The `existing_capacity` parameter defines the capacity installed prior to the beginning of `time_optimize`. Note that processes with existing capacity that would survive into future periods require all of the engineering-economic characteristics of a standard process, with the exception of an investment cost.

global_discount_rate

GDR

The `global_discount_rate` parameter represents the global discount rate used to convert cash flows in future model time periods into a present value. The net present value (NPV) of a cashflow is related to its future value (FV) via the formula:

$$\text{NPV} = \frac{\text{FV}}{(1+GDR)^n}$$

where n is in years. This parameter is used to calculate all discounted costs, which are the basis of the objective function. Costs are discounted to the first future time period in the model.

The output in the `output_cost` table shows both discounted and non-discounted (raw) values for all model costs.

The `global_discount_rate` is entered in the metadata_real table in the database.

loan_lifetime_process

$LLP_{r \in R, t \in T, v \in P}$

Temoa gives the modeler the ability to separate the loan lifetime from the useful life of a process. This parameter specifies the loan term associated with capital investment in a process, in years. If not specified, the model assigns the technology lifetime to the loan period.

lifetime_process

$LTP_{r \in R, t \in T, v \in P}$

This parameter specifies the total useful life of a given process in years.

lifetime_survival_curve

$LSC_{r \in R, p \in P, t \in T, v \in V}$

The `lifetime_survival_curve` parameter represents the surviving fraction of original capacity for a given process at a given period. It allows the modeler to specify gradual capacity loss over time rather than abrupt retirement at end of life. Values should be between 0 and 1, where 1 means full survival.

lifetime_tech

$LTT_{r \in R, t \in T}$

Similar to `lifetime_process`, this parameter specifies the total useful life of a given technology in years, with a default of 40 years. If all vintages of a given technology have the same lifetime, then `lifetime_tech` can be used instead of `lifetime_process`.

linked_techs

$LIT_{r \in R, t \in T, e \in C^e, t' \in T}$

In power-to-gas pathways, $CO2$ is an input to some processes, including synthetic natural gas production and liquid fuel production via Fischer-Tropsch. Within the model, $CO2$ must be converted from an emissions commodity to a physical commodity that can be included in the `efficiency` table. The `linked_techs` parameter specifies the dummy technology used to convert an emissions commodity to a physical commodity. Note that the first `t` represents the primary upstream technology linked to the dummy linked technology, which is represented by the second `t'` index.

loan_rate

$LR_{r \in r, t \in T, v \in V}$

The interest rate used for loans supporting investment costs. The default loan rate is accessible in the metadata_real table in the database.

default_loan_rate

DLR

A scalar parameter specifying the default interest rate applied to loans when no process-specific `loan_rate` is defined. This value is read from the `metadata_real` table in the database.

limit_activity

$LA_{r \in R, p \in P, t \in T}$

The `limit_activity` parameter places an upper or lower bound on the total activity (energy production) from a technology or technology group in each model time period. The bound direction is controlled by the `operator` column (`le`, `ge`, or `eq`). The `tech_or_group` column accepts either a single technology name or a group name defined in `tech_group_names`.

limit_activity_share

$$LAS_{r \in R, p \in P, g_1, g_2}$$

The `limit_activity_share` parameter constrains the activity of one technology or group as a share of another technology or group's activity. For example, it can enforce a minimum renewable generation share. The operator column controls whether the share is an upper bound, lower bound, or equality.

limit_annual_capacity_factor

$$LACF_{r \in R, t \in T, v \in V, o \in C_c}$$

The `limit_annual_capacity_factor` parameter limits the annual capacity factor of a process — that is, the ratio of actual annual output to maximum possible output. The `tech_or_group` column accepts a technology name or group name. Note this is indexed by vintage, not period, and will be applied to all valid time periods for that process.

limit_capacity

$$LC_{r \in R, p \in P, t \in T}$$

The `limit_capacity` parameter places an upper or lower bound on the total available (retirement-adjusted) capacity of a technology or technology group in each model timeperiod. The `operator` column controls the bound direction. The `tech_or_group` column accepts either a single technology name or a group name.

limit_capacity_share

$$LCS_{r \in R, p \in P, g_1, g_2}$$

The `limit_capacity_share` parameter constrains the capacity of one technology group as a share of another group's capacity. The operator column controls the bound direction. The group columns accept either a single technology name or a technology group name.

limit_emission

$$LE_{r \in R, p \in P, e \in C^e}$$

The `limit_emission` parameter ensures that Temoa finds a solution that fits within the modeler-specified limit on emission e in time period p. The operator column controls whether this is an upper bound, lower bound, or equality.

limit_new_capacity

$LNC_{r \in R, t \in T, v \in V}$

The `limit_new_capacity` parameter constrains the amount of new capacity that can be deployed for a given technology or group in a specific vintage. The `tech_or_group` column accepts a technology name or group name.

limit_new_capacity_share

$LNCS_{r \in R, g_1, g_2, v \in V}$

The `limit_new_capacity_share` parameter constrains the new capacity of one technology or group as a share of another technology or group's new capacity.

limit_resource

$LS_{r \in R, t \in T}$

The `limit_resource` parameter represents a bound on the cumulative amount of commodity that can be produced by a technology or group over the entire model time horizon. The `tech_or_group` column accepts a technology name or group name. Note that this is *not* supported in myopic mode as the cumulative limit would need to decline as the horizon moves forward, which has not been added yet.

limit_seasonal_capacity_factor

$LSCF_{r \in R, s \in S, t \in T}$

The `limit_seasonal_capacity_factor` parameter limits the capacity factor of a technology in a specific season. There is no period index: the limit applies across all periods. The `tech_or_group` column accepts a technology name or group name.

limit_storage_fraction

$LSF_{r \in R, s \in S, d \in D, t \in T^S}$

The `limit_storage_fraction` parameter constrains the storage level of a storage technology in any time slice to be at or above a specified fraction of its maximum charge. Values should be between 0 and 1.

Note that this constraint will be applied to all valid `period` and `vintage` combinations for the technology.

limit_tech_input_split

$TIS_{r \in R, p \in P, i \in C_p, t \in T}$

Some technologies have a single output but have multiple input fuels. The `limit_tech_input_split` parameter constrains the shares of commodity input to a specific technology in a given period in every time slice. The `operator` column controls whether the share is an upper bound, lower bound, or equality.

limit_tech_input_split_annual

$TISA_{r \in R, p \in P, i \in C_p, t \in T}$

Similar to `limit_tech_input_split`, but constrains the average input commodity shares at the annual level, allowing the shares at the time slice level to vary.

limit_tech_output_split

$TOS_{r \in R, p \in P, t \in T, o \in C_c}$

Some technologies have a single input fuel but have multiple outputs. The `limit_tech_output_split` parameter constrains the shares of commodity output from a specific technology in a given period by time slice.

limit_tech_output_split_annual

$TOSA_{r \in R, p \in P, t \in T, o \in C_c}$

Similar to `limit_tech_output_split`, but constrains the average output commodity shares at the annual level, allowing the shares at the time slice level to vary.

planning_reserve_margin

$PRM_{r_g \in R, t_g \in T}$

The required excess of credited installed capacity above demand, expressed as a fraction of demand, keyed by a region-or-group r_g and a technology-or-group t_g. For example, a value of 0.2 requires that credited capacity be at least 120% of demand. Demand is estimated from production by time slice. When a region group is used, any exchange region (e.g. `r1-r2`) where exactly one endpoint belongs to

the group is automatically included in the reserve calculation; this auto-inclusion is unique to the reserve margin constraints.

planning_reserve_credit

$PRC_{r \in R, t \in T}$

The fraction of a technology's installed capacity that can be reliably counted toward the reserve margin. A firm, fully dispatchable process (e.g. a gas turbine) typically receives a credit near 1, while a weather-dependent process (e.g. wind or solar) receives a lower value.

operating_reserve_margin

$ORM_{r_g \in R, t_g \in T}$

The dynamic counterpart to `planning_reserve_margin`, indexed the same way by region-or-group and technology-or-group. Rather than crediting installed capacity, it requires that available (derated) generation in each time slice exceed the region-group's proxy demand by this margin. When a region group is used, any exchange region (e.g. `r1-r2`) where exactly one endpoint belongs to the group is automatically included in the reserve calculation; this auto-inclusion is unique to the reserve margin constraints.

operating_reserve_derate

$ORD_{r \in R, s \in S, t \in T}$

The fraction of a technology's available output that can be depended upon in a given season. A value less than 1 reflects the fact that not all of a process's capacity is reliably available — for example, due to scheduled maintenance or seasonal resource constraints. Defaults to 1.

ramp_down_hourly

$RDH_{r \in R, t \in T}$

The `ramp_down_hourly` parameter specifies the fraction of installed capacity by which a technology can ramp output down per hour. This is used in the `ramp_down_constraint`.

ramp_up_hourly

$RUH_{r \in R, t \in T}$

The `ramp_up_hourly` parameter specifies the fraction of installed capacity by which a technology can ramp output up per hour. This is used in the `ramp_up_constraint`.

segment_fraction

$SEG_{s \in S, d \in D}$

The `segment_fraction` parameter specifies the fraction of the year represented by each combination of season and time of day. In v4, this parameter is **computed automatically** from two user-specified inputs:

> - `segment_fraction` column in the `time_season` table — the fraction of the year each season represents (e.g. 0.25 for each quarter).
> - `hours` column in the `time_of_day` table — the number of hours each time-of-day segment represents (e.g. 8 for "Day", 16 for "Night").

The computation is:

$$SEG_{s,d} = \text{segment\_fraction\_per\_season}(s) \times \frac{\text{hours}(d)}{\sum_{d'} \text{hours}(d')}$$

The sum of all $SEG_{s,d}$ values must equal 1, representing 100% of a year.

segment_fraction_per_season

$SFS_{s \in S}$

The per-season share of the year, loaded directly from the `segment_fraction` column of the `time_season` database table. These values are the first factor in the segment_fraction computation and must sum to 1.

segment_fraction_per_sequential_season

$SFS_{s^{seq} \in S^{seq}}$

The per-sequential-season share of the year, loaded from the `segment_fraction` column of the `time_season_sequential` database table. Used to compute the `days_adjust` ratio that rescales storage levels and flows between non-sequential and sequential season representations.

storage_duration

$SD_{r \in R, t \in T^S}$

The `storage_duration` parameter represents the number of hours over which storage can discharge if it starts at full charge and produces maximum output until empty. The parameter value defaults to 4 hours if not specified by the user.

*loan_annualize

$LA_{r \in R, t \in T, v \in P}$

This is a model-calculated parameter based on the process-specific loan length (its indices are the same as the `loan_lifetime_process` parameter), and process-specific interest rate (the `loan_rate` parameter). It is calculated via the formula:

$$LA_{t,v} = \frac{DR_{r,t,v}}{1 - (1 + DR_{r,t,v})^{-LLN_{r,t,v}}} \quad \forall \{t, v\} \in \Theta_{\text{cost\_invest}}$$

*period_length

$LEN_{p \in P}$

Given that the modeler may specify arbitrary time period boundaries, this parameter specifies the number of years contained in each period. The final year is the largest element in `time_future` which is specifically not included in the list of periods in `time_optimize` (P^o). The length calculation for each period then exploits the fact that the `time` sets are ordered:

$$\begin{aligned} \text{LET boundaries} &= \text{sorted}(\text{P}^f) \\ \text{LET I(p)} &= \text{index of p in boundaries} \\ &\therefore \\ LEN_p &= \text{boundaries}[I(p) + 1] - p \quad \forall p \in P \end{aligned}$$

The first line creates a sorted array of the period boundaries, called *boundaries*. The second line defines a function *I* that finds the index of period p in boundaries. The third line then defines the length of period p to be the number of years between period p and the next period. For example, if $\text{P}^f = \{2015, 2020, 2030, 2045\}$, then *boundaries* would be `[2015, 2020, 2030, 2045]`. For 2020, I(2020) would return 2. Similarly, boundaries[3] = 2030. Then,

$$
\begin{aligned}
LEN_{2020} &= \text{boundaries}[I(2020)+1] - (2020) \\
&= \text{boundaries}[2+1] - 2020 \\
&= \text{boundaries}[3] - 2020 \\
&= 2030 - 2020 \\
&= 10
\end{aligned}
$$

Note that LEN is only defined for elements in P^o, and is specifically not defined for the final element in P^f.

*process_life_frac

$PLF_{r \in R, p \in P, t \in T, v \in P}$

The modeler may specify a useful lifetime of a process such that the process will be decommissioned part way through a period. Rather than attempt to delineate each year within that final period, Temoa averages the total output of the process over the entire period but limits the available capacity and output of the decommissioning process by the ratio of how long through the period the process is active. This parameter is that ratio, formally defined as:

$$
PLF_{p,t,v} = \frac{v + LTP_{t,v} - p}{LEN_p}
$$

$$
\begin{aligned}
&\forall \{p,t,v\} \in \Theta_{\text{Activity by PTV}} | \\
&v + LTP_{t,v} \notin P, \\
&v + LTP_{t,v} \leq max(F), \\
&p = max(P | p < v + LTP_{t,v})
\end{aligned}
$$

Note that this parameter is defined over the same indices as `cost_variable` – the active periods for each process $\{p,t,v\}$. As an example, if a model has $P = \{2010, 2012, 2020, 2030\}$, and a process $\{t,v\} = \{car, 2010\}$ has a useful lifetime of 5 years, then this parameter would include only the first two activity indices for the process. Namely, $p \in \{2010, 2012\}$ as $\{p,t,v\} \in \{\{2010, car, 2010\}, \{2012, car, 2010\}\}$. The values would be $TLF_{2010,car,2010} = 1$, and $TLF_{2012,car,2010} = \frac{3}{8}$.

Decision Variables

Decision variables are the quantities optimized by the model. A summary table of decision variables is provided below, followed by a more detailed description of each.

Temoa's Main Variables

Variable	Temoa Name	Short Description
$FO_{r,p,s,d,i,t,v,o}$	`v_flow_out`	Commodity flow by time slice out of a tech based on a given input
$FOA_{r,p,i,t,v,o}$	`v_flow_out_annual`	Annual commodity flow out of a tech based on a given input
$FIS_{r,p,s,d,i,t,v,o}$	`v_flow_in`	Commodity flow into a storage tech to produce a given output
$FLX_{r,p,s,d,i,t,v,o}$	`v_flex`	The portion of commodity production exceeding demand
$FLXA_{r,p,i,t,v,o}$	`v_flex_annual`	The portion of commodity production from constant production techs exceeding demand
$CUR_{r,p,s,d,i,t,v,o}$	`v_curtailment`	Commodity flow out of a tech that is curtailed
$CAP_{r,p,t,v}$	`v_capacity`	Required tech capacity to support associated activity
$CAPAVL_{r,p,t}$	`v_capacity_available_by_period_and_tech`	Derived variable representing the capacity of technology t available in period p
$SI_{r,p,s,t,v}$	`v_storage_init`	Hub variable for the initial charge level of each daily storage cycle
$SL_{r,p,s,d,t,v}$	`v_storage_level`	Charge level each time slice associated with storage techs
$SSL_{r,p,s,t,v}$	`v_seasonal_storage_level`	Base charge level of sequential seasons for seasonal storage
$RCAP_{r,p,t,v}$	`v_retired_capacity`	Capacity retired before end of life
$ART_{r,p,t,v}$	`v_annual_retirement`	Annualised capacity retiring or reaching end of life
$NCAP_{r,t,v}$	`v_new_capacity`	New deployed capacity

v_flow_out

$FO_{r,p,s,d,i,t,v,o}$

The most fundamental variable in the Temoa formulation is the `v_flow_out` variable. It describes the commodity flow out of a process in a given time slice. To balance input and output flows in the `CommodityBalance_constraint`, the commodity flow into a given process can be calculated as $\sum_{T,V,O} \mathbf{FO}_{p,s,d,c,t,v,o} / EFF_{c,t,v,o}$.

v_flow_out_annual

$FOA_{r,p,i,t,v,o}$

Similar to `v_flow_out`, but used for technologies that are members of the `tech_annual` set, whose output does not vary across seasons and times-of-day. Eliminating the `s,d` indices for these technologies improves computational performance.

v_flex

$FLX_{r,p,s,d,i,t,v,o}$

In some cases, the overproduction of a commodity may be required, such that supply exceeds the endogenous demand. Refineries represent a common example, where the share of different refined products are governed by tech_output_split, but total production is driven by a particular commodity. For example, gasoline production may be artificially constrained in order to ensure the appropriate balance for lower demand fuels such as propane or kerosene. Instead, we allow overproduction, i.e., production exceeding endogenous demand, for commodities produced by technologies belonging to the `tech_flex` set. In the example above, adding the refinery to the `tech_flex` set allows for the overproduction of propane and kerosene, allowing the model to fulfill the endogenous demand for gasoline. This flexible technology designation activates a slack variable ($\mathbf{FLX}_{r,p,s,d,i,t,v,c}$)representing the excess production in the `CommodityBalanceAnnual_constraint`.

v_flex_annual

$FLXA_{r,p,i,t,v,o}$

Similar to `v_flex`, but used for technologies that are members of the `tech_flex` set, whose output does not vary across seasons and times-of-day. Eliminating the `s,d` indices for these technologies improves computational performance.

v_curtailment

$CUR_{r,p,s,d,i,t,v,o}$

The `v_curtailment` variable is an accounting tool to help calculate the unused production capacity of technologies annotated in the Technology database table as curtailable technologies belonging to the `tech_curtailment` set. Renewables such as wind and solar are often placed in this set. While we used to simply formulate the `Capacity` and `CommodityBalance` constraints as inequalities that implicitly allowed for curtailment, this simpler approach does not work with renewable targets because the curtailed portion of the electricity production counts towards the target, and there is no way to distinguish it from the useful production. Including an explicit curtailment term addresses the issue. Curtailment in the model is simply the production activity that is not used in the model and is reported as such in the output_curtailment table. Note: Outputs presented in the *output_curtailment* table for curtailment (the table separately includes flex outputs) are limited by Capacity Factor. Meaning: if a tech has a capacity of 10 units, and a CF of 0.8 and a usage of 5 units, then the reported curtailment is 3 units (0.8 x 10 - 5).

v_flow_in_storage

$FIS_{r,p,s,d,i,t,v,o}$

Because the production and consumption associated with storage techs occur across different time slices, the comodity flow into a storage technologiy cannot be discerned from `v_flow_out`. Thus an explicit $flow_in$ variable is required for storage.

v_capacity

$CAP_{r,p,t,v}$

The `v_capacity` variable represents the available capacity of a process (r, t, v) in period p, adjusted for retirements or end of life. Temoa constrains the capacity variable to be able to meet the total commodity flow out of that process in all time slices in which it is active (1).

v_capacity_available_by_period_and_tech

$CAPAVL_{r,p,t}$

`CapacityAvailableByPeriodAndTech` is a convenience variable that is not strictly necessary, but used where the individual vintages of a technology are not warranted (e.g. in calculating the maximum or minimum total capacity allowed in a given time period).

v_storage_init

$SI_{r,p,s,t,v}$

The `v_storage_init` variable replaces the `v_storage_level` variable for the initial storage charge level of each season. This is purely for solver optimisation and has no impact on model structure or results. This 1:1 swap measurably improves presolve time in large models with storage. The mechanism behind this is not understood.

v_storage_level

$SL_{r,p,s,d,t,v}$

The `v_storage_level` variable tracks the storage charge level across ordered time slices and is critical to ensure that storage charge and dispatch is constrained by the energy available in the storage units.

v_seasonal_storage_level

$SSL_{r,p,s,t,v}$

The `v_seasonal_storage_level` variable tracks the base charge level at the boundary between sequential seasons for technologies flagged as seasonal storage (`tech_seasonal_storage`). It is used in the `seasonal_storage_energy_constraint` to chain the storage state of charge across seasons defined in the `time_season_sequential` table.

v_retired_capacity

$RCAP_{r,p,t,v}$

The `v_retired_capacity` variable tracks the cumulative capacity of a process that has been retired before its natural end of life, up to and including period p. Only technologies in the `tech_retirement` set may retire early in this manner. This variable is non-decreasing across periods.

v_annual_retirement

$ART_{r,p,t,v}$

The `v_annual_retirement` variable represents the annualised capacity that retires or reaches end of life in period p. It accounts for both early retirements (via `v_retired_capacity`) and natural end-of-life.

v_new_capacity

$NCAP_{r,t,v}$

The `v_new_capacity` variable represents the newly deployed capacity of a technology in its vintage period. It is the primary investment decision variable and drives the capital cost terms in the objective function. Unlike `v_capacity`, it has no period index — capacity is built once in its vintage year.

We explain the equations governing these variables in the Equations section.

Equations

There are four main equations that govern the flow of energy through the model network. The `Demand_Constrant` (6) ensures that the supply meets demand in every time slice. For each process, the `Capacity_constraint` (1) ensures that there is sufficient capacity to meet the optimal commodity flows across all time slices. Between processes, the `CommodityBalance_constraint` (7) ensures that global commodity production across the energy system is sufficient to meet the endogenous demands for that commodity. Finally, the objective function (25) drives the model to minimize the system-wide cost of energy supply by optimizing the deployment and utilization of energy technologies across the system.

One additional point regarding the model formulation. Technologies that produce constant annual output can be placed in the `tech_annual` set. While not required, doing so improves computational performance by eliminating the season and time of day `(s,d)` indices associated with these technologies. In order to ensure the model functions correctly with these simplified technologies, slightly different formulations of the capacity and commodity balance constraints are required. See the `AnnualCommodityBalance_constraint` and `CapacityAnnual_constraint` (2) below for details.

The rest of this section defines each model constraint, with a rationale for existence. We use the implementation-specific names for the constraints to highlight the organization of the functions within the actual code. Note that the definitions below are pulled directly from the docstrings embedded in `temoa/components/`.

A couple notes on the notation below. First, in all equations, we **bold** variables to distinguish them from parameters. Second, you will notice the use of the Theta superset (Θ). The Temoa code makes heavy use of sparse sets, for both correctness and efficient use of computational resources. For brevity, and to avoid discussion of implementation details, we do not enumerate their logical creation here. Instead, we rely on the reader's general understanding of the context. The use of (Θ) means that the constraint is only defined for the exact indices that the modeler specified.

Capacity-Defining Constraints

We begin with the `Capacity_constraint` and `CapacityAnnual_constraint`, which are particularly important because they define the relationship between installed capacity and allowable commodity flow.

temoa.components.capacity.capacity_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

This constraint ensures that the capacity of a given process is sufficient to support its activity across all time periods and time slices. The calculation on the left hand side of the equality is the maximum amount of energy a process can produce in the timeslice `(s,d)`. Note that the curtailment variable shown below only applies to technologies that are members of the curtailment set. Curtailment is necessary to track explicitly in scenarios that include a high renewable target. Without it, the model can generate more activity than is used to meet demand, and have all activity (including the portion curtailed) count towards the target. Tracking activity and curtailment separately prevents this possibility.

$$(\text{CFP}_{r,s,d,t,v} \cdot \text{C2A}_{r,t} \cdot \text{SEG}_{s,d}) \cdot \mathbf{CAP}_{r,p,t,v} = \sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} + \sum_{I,O} \mathbf{CUR}_{r,p,s,d,i,t,v,o} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{FO}} \tag{1}$$

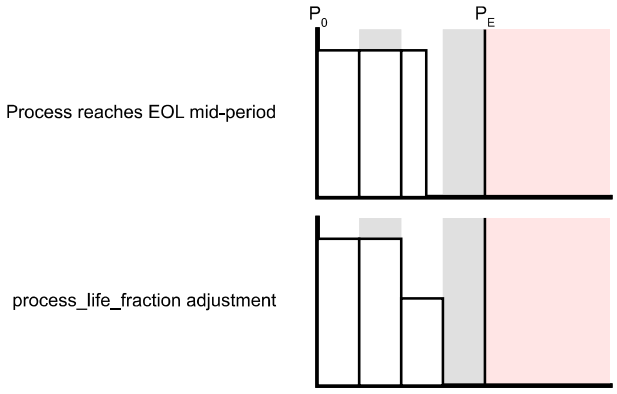
temoa.components.capacity.capacity_annual_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage*) → ExprLike [source]

Similar to Capacity_constraint, but for technologies belonging to the `tech_annual` set. Technologies in the tech_annual set have constant output across different timeslices within a year, so we do not need to ensure that installed capacity is sufficient across all timeslices, thus saving some computational effort. Instead, annual output is sufficient to calculate capacity. Hourly capacity factors cannot be defined to annual technologies but annual capacity factors can be set using limit_annual_capacity_factor, which will be implicitly accounted for here.

$$\text{C2A}_{r,t} \cdot \mathbf{CAP}_{r,p,t,v} \geq \sum_{I,O} \mathbf{FOA}_{r,p,i,t \in T^a,v,o} \quad \forall \{r,p,t \in T^a,v\} \in \Theta_{\text{Activity}} \tag{2}$$

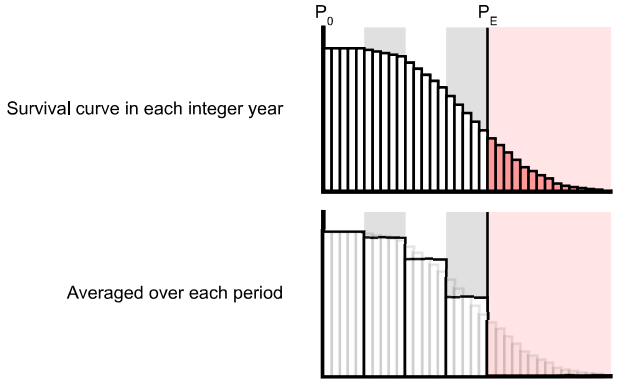
temoa.components.capacity.adjusted_capacity_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage*) → ExprLike [source]

This constraint updates the capacity of a process by taking into account retirements and end of life. For a given `(r,p,t,v)` index, this constraint sets the capacity equal to the amount installed in period `v` and subtracts from it any and all retirements that occurred prior to the period in question, `p`, and end of life from the survival curve if defined. It finally adjusts for the process life fraction, which accounts for a possible mid-period end of life where, for example, EOL 3 years into a 5-year period would be treated as $\frac{3}{5}$ capacity for all 5 years.



For processes reaching end of life mid-period, the process life fraction adjustment is applied, distributing the effective capacity over the whole period.

For processes using survival curves, the yearly survival curve $\mathrm{LSC}_{r,p,t,v}$ is averaged over the period to get the effective remaining capacity for that period Because this implicitly handles mid-period end of life, $\mathrm{PLF}_{r,p,t,v}$ is used to account for both phenomena.



For processes with a defined survival curve, the surviving capacity is averaged over each period to get the adjusted capacity. This implicitly handles mid-period end of life as a survival curve will always be zero after the end of life of a process.

$$\mathbf{CAP}_{r,p,t,v} = \begin{cases} \mathrm{PLF}_{r,p,t,v} \cdot \left(\mathrm{ECAP}_{r,t,v} - \sum\limits_{v<p'<=p} \frac{\mathbf{RCAP}_{r,p',t,v}}{\mathrm{LSC}_{r,p',t,v}} \right) & \text{if } \; v \in T^e \\ \mathrm{PLF}_{r,p,t,v} \cdot \left(\mathbf{NCAP}_{r,t,v} - \sum\limits_{v<p'<=p} \frac{\mathbf{RCAP}_{r,p',t,v}}{\mathrm{LSC}_{r,p',t,v}} \right) & \text{if } \; v \notin T^e \end{cases} \tag{3}$$

$$\text{where } \mathrm{PLF}_{r,p,t,v} = \begin{cases} \frac{1}{\mathrm{LEN}_p} \cdot \left(\sum\limits_{y=p}^{p+\mathrm{LEN}_p-1} \mathrm{LSC}_{r,y,t,v} \right) & \text{if } t \in T^{sc} \\ \frac{1}{\mathrm{LEN}_p} \cdot (v + \mathrm{LTP}_{r,t,v} - p) & \text{if } t \notin T^{sc} \end{cases}$$

We divide $\frac{\mathbf{RCAP}_{r,p',t,v}}{\mathrm{LSC}_{r,p',t,v}}$ because the average survival factor in $\mathrm{PLF}_{r,p,t,v}$ is indexed to the vintage period (the beginning of the survival curve). So, we adjust for the relative survival from the time when that retirement occurred (treated here as at the beginning of each period).

temoa.components.capacity.annual_retirement_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage*) → ExprLike [source]

Print to PDF

Get the annualised retirement rate for a process in a given period. Used to output retirement (including end of life, EOL) and to model end of life flows and emissions. Assumes that retirement from the beginning of each period is evenly distributed over that model period $\frac{1}{\mathrm{LEN}_p}$ for the accounting of retirement flows (in the same way we assume capacity is deployed evenly over the model period for construction inputs and embodied emissions). The factor $\frac{\mathrm{LSC}_{r,p,t,v}}{\mathrm{PLF}_{r,p,t,v}}$ adjusts the average survival during a period to the survival at the beginning of that period.

$$\mathbf{ART}_{r,p,t,v} = \begin{cases} \frac{1}{\mathrm{LEN}_p} \cdot \frac{\mathrm{LSC}_{r,p,t,v}}{\mathrm{PLF}_{r,p,t,v}} \cdot \mathbf{CAP}_{r,p,t,v} & \text{if EOL} \\ \frac{1}{\mathrm{LEN}_p} \cdot \left(\frac{\mathrm{LSC}_{r,p,t,v}}{\mathrm{PLF}_{r,p,t,v}} \cdot \mathbf{CAP}_{r,p,t,v} - \frac{\mathrm{LSC}_{r,p_{next},t,v}}{\mathrm{PLF}_{r,p_{next},t,v}} \cdot \mathbf{CAP}_{r,p_{next},t,v} \right) & \text{otherwise} \end{cases} \tag{4}$$

$$\text{where EOL when } p \leq v + LTP_{r,t,v} < p + LEN_p$$

temoa.components.capacity.capacity_available_by_period_and_tech_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology*) → ExprLike [source]

The $\mathbf{CAPAVL}$ variable is nominally for reporting solution values, but is also used in the Limit constraint calculations.

$$\mathbf{CAPAVL}_{r,p,t} = \sum_{v,p_i \leq p} \mathbf{CAP}_{r,p,t,v} \tag{5}$$

$$\forall p \in \mathrm{P}^o, r \in R, t \in T$$

Network Constraints

These three constraints define the core of the Temoa model; together, they define the algebraic energy system network.

temoa.components.commodities.demand_constraint(*model: TemoaModel, r: Region, p: Period, dem: Commodity*) → ExprLike [source]

The Demand constraint drives the model. This constraint ensures that supply meets the demand specified by the Demand parameter in all periods, by ensuring that the sum of all the annual demand output commodity (dem) generated by $\mathbf{FOA}$ across all technologies and vintages equals the modeler-specified demand.

$$\sum_{I,T,V} \mathbf{FOA}_{r,p,i,t,v,dem} = DEM_{r,p,dem} \tag{6}$$

The per-timeslice distribution of demand across non-annual technologies is enforced by the separate `demand_activity_constraint`. In the case of a singleton demand, where only one `(r, i, t, v)` sub-process produces the demand commodity, the flow variables are fixed directly and demand constraints are skipped.

Note that the validity of this constraint relies on the fact that the C^d set is distinct from both C^e and C^p. In other words, an end-use demand must only be an end-use demand. Note that if an output could satisfy both an end-use and internal system demand, then the output from $\mathbf{FO}$ and $\mathbf{FOA}$ would be double counted.

temoa.components.commodities.commodity_balance_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, c: Commodity*) → ExprLike [source]

Where the Demand constraint (6) ensures that end-use demands are met, the CommodityBalance constraint ensures that the endogenous system demands are met. This constraint requires the total production of a given commodity to equal the amount consumed, thus ensuring an energy balance at the system level. In this most general form of the constraint, the energy commodity being balanced has variable production at the time slice level. The energy commodity can then be consumed by three types of processes: storage technologies, non-storage technologies with output that varies at the time slice level, and non-storage technologies with constant annual output.

Separate expressions are required in order to account for the consumption of commodity c by downstream processes. For the commodity flow into storage technologies, we use $\mathbf{FI}_{r,p,s,d,i,t,v,c}$. Note that the FlowIn variable is defined only for storage technologies, and is required because storage technologies balance production and consumption across time slices rather than within a single time slice. For commodity flows into non-storage processes with time varying output, we use $\mathbf{FO}_{r,p,s,d,i,t,v,c}/EFF_{r,i,t,v,o}$. The division by $EFF_{r,c,t,v,o}$ is applied to the output flows that consume commodity c to determine input flows. Finally, we need to account for the consumption of

commodity c by the processes in `tech_annual`. Since the commodity flow of these processes is on an annual basis, we use $SEG_{s,d}$ to calculate the consumption of commodity c in time-slice (s, d) from the annual flows. Formulating an expression for the production of commodity c is more straightforward, and is simply calculated by $\mathbf{FO}_{r,p,s,d,i,t,v,c}$.

In some cases, the overproduction of a commodity may be required, such that the supply exceeds the endogenous demand. Refineries represent a common example, where the share of different refined products are governed by TechOutputSplit, but total production is driven by a particular commodity like gasoline. Such a situation can result in the overproduction of other refined products, such as diesel or kerosene. In such cases, we need to track the excess production of these commodities. To do so, the technology producing the excess commodity should be added to the `tech_flex` set. This flexible technology designation will activate a slack variable ($\mathbf{FLX}_{r,p,s,d,i,t,v,c}$) representing the excess production in the `CommodityBalanceAnnual_constraint`. Note that the `tech_flex` set is different from `tech_curtailment` set; the latter is technology- rather than commodity-focused and is used in the `Capacity_constraint` to track output that is used to produce useful output and the amount curtailed, and to ensure that the installed capacity covers both. Alternatively, the commodity can be added to the `commodity_waste` set, for which this equality constraint becomes an inequality constraint, allowing production to exceed consumption for a single commodity.

This constraint also accounts for imports and exports between regions when solving multi-regional systems. The import ($\mathbf{FIM}$) and export ($\mathbf{FEX}$) variables are created on-the-fly by summing the $\mathbf{FO}$ variables over the appropriate import and export regions, respectively, which are defined in `temoa_initialize.py` by parsing the `tech_exchange` processes.

Consumption of the commodity by construction inputs is annualised using the period length. Production of the commodity by end-of-life outputs uses the AnnualRetirement variable, which is already annualised.

Finally, for annual commodities, AnnualCommodityBalance is used which balances the sum of flows over each year.

process outputs + imports + end of life outputs = process inputs + construction inputs + exports + flex waste

$$
\begin{aligned}
&\sum_{I,t\notin T^a,V} \mathbf{FO}_{r,p,s,d,i,t,v,c} && \text{(processes outputting commodity)}\\
&+ SEG_{s,d} \cdot \sum_{I,t\in T^a,V} \mathbf{FOA}_{r,p,i,t,v,c} && \text{(annual processes outputting commo}\\
&+ \sum_{\text{reg}\neq r,I,t\in T^x,V} \mathbf{FIM}_{r-\text{reg},p,s,d,i,t,v,c} && \text{(inter-regional imports of commodity}\\
&+ SEG_{s,d} \sum_{T,V} (EOLO_{r,t,v,c} \cdot \mathbf{ART}_{r,p,t,v}) && \text{(end-of-life outputs of commodity)}\\
&\begin{cases} = \text{if } c \notin C^w \\ \geq \text{if } c \in C^w \end{cases} &&\\
&\sum_{t\in T^s,V,O} \mathbf{FIS}_{r,p,s,d,c,t,v,o} && \text{(commodity stored)}\\
&+ \sum_{t\notin T^s,V,O} \frac{\mathbf{FO}_{r,p,s,d,c,t,v,o}}{EFF_{r,c,t,v,o}} && \text{(commodity consumed by processes)}\\
&+ SEG_{s,d} \cdot \sum_{t\in T^a,V,O} \frac{\mathbf{FOA}_{r,p,c,t,v,o}}{EFF_{r,c,t,v,o}} && \text{(commodity consumed by annual prc}\\
&+ \sum_{\text{reg}\neq r,t\in T^x,V,O} \mathbf{FEX}_{r-\text{reg},p,s,d,c,t,v,o} && \text{(inter-regional exports of commodity}\\
&+ \sum_{I,t\in T^f,V} \mathbf{FLX}_{r,p,s,d,i,t,v,c} && \text{(flex wastes of commodity)}\\
&+ SEG_{s,d} \cdot \sum_{T,V} \left(CON_{r,c,t,v} \cdot \frac{\mathbf{NCAP}_{r,t,v}}{LEN_p} \right) && \text{(consumed annually by construction}\\
& && \forall \{r,p,s,d,c\} \in \Theta_{\text{Commodi}}
\end{aligned}
\tag{7}
$$

temoa.components.commodities.annual_commodity_balance_constraint(*model: TemoaModel, r: Region, p: Period, c: Commodity*) → ExprLike [source]

Similar to CommodityBalance_constraint but only balances the supply and demand of the commodity at the period level, summing all flows over the period but allowing imbalances at the time slice or seasonal level. Applies only to commodities in the `commodity_annual` set.

Physical and Operational Constraints

These constraints fine-tune the model formulation to account for various physical and operational real-world phenomena.

temoa.components.operations.baseload_diurnal_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike

Some electric generators cannot ramp output over a short period of time (e.g., hourly or daily). Temoa models this behavior by forcing technologies in the `tech_baseload` set to maintain a constant output across all times-of-day within the same season. Note that the output of a baseload process can vary between seasons. [source]

Ideally, this constraint would not be necessary, and baseload processes would simply not have a d index. However, implementing the more efficient functionality is currently on the Temoa TODO list.

$$SEG_{s,D_0} \cdot \sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} = SEG_{s,d} \cdot \sum_{I,O} \mathbf{FO}_{r,p,s,D_0,i,t,v,o} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{BaseloadDiurnal}} \tag{8}$$

temoa.components.commodities.demand_activity_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage, dem: Commodity*) → ExprLike [source]

For end-use demands, it is unreasonable to let the model arbitrarily shift the use of demand technologies across time slices. For instance, if household A buys a natural gas furnace while household B buys an electric furnace, then both units should be used throughout the year. Without this constraint, the model might choose to only use the electric furnace during the day, and the natural gas furnace during the night.

This constraint ensures that the ratio of a process activity to demand is constant for all time slices. In other words, while the model may choose how each technology contributes to a demand at the annual level, the time slice level distribution of the technology's contribution to the demand is fixed to the demand specific distribution.

$$\sum_{I} \mathbf{FOA}_{r,p,i,t,v,dem} \cdot \text{DSD}_{r,s,d,dem} = \sum_{I} \mathbf{FO}_{r,p,s,d,i,t,v,dem} \quad \forall \{r,p,s,d,t,v,dem\} \in \Theta_{\text{DemandActivity}} \tag{9}$$

Note that this constraint is unnecessary and therefore skipped for annual technologies or singleton demands.

temoa.components.storage.storage_energy_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

This constraint enforces the continuity of storage level between time slices.

Uses the `time_next` mapping to chain storage levels forward. For non-seasonal storage, $v\_storage\_init$ replaces $v\_storage\_level$ on the LHS at the daily cycle wrap point (last time-of-day). Combined with the tie-back constraint ($SL_{d_{last}} = SI$), this is structurally equivalent to a closed cycle.

Empirically, this swap improved barrier solve time ~25% on a 16-region national model, though the structural reason is not fully understood.

Non-seasonal, last time-of-day (d_last):

$$SI_{r,p,s,t,v} + \text{net\_charge} = SL_{r,p,s_{next},d_{next},t,v}$$

All other time slices (non-seasonal and seasonal):

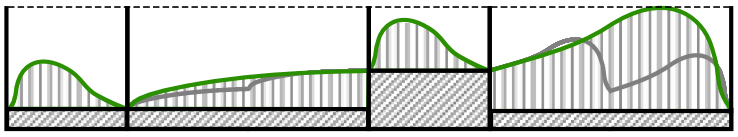
$$SL_{r,p,s,d,t,v} + \text{net\_charge} = SL_{r,p,s_{next},d_{next},t,v}$$

For seasonal storage, the last time-of-day is skipped (handled by SeasonalStorageEnergy_constraint).

`temoa.components.storage.`**`seasonal_storage_energy_constraint`**`(`*`model: TemoaModel, r: Region, p: Period, s_seq: Season, t: Technology, v: Vintage`*`)` → `ExprLike` [source]

This constraint enforces the continuity of state of charge between seasons for seasonal storage. Sequential season storage level increases by the matched season's net charge over that entire day, adjusted for number of days represented by sequential vs non-sequential seasons. Only applies to storage technologies in the `tech_seasonal_storage` set. s^* represents the matching non-sequential season for the sequential season s^{seq}.

$$\mathbf{SSL}_{r,p,s^{seq},t,v} + \frac{\text{SFS}_{s^{seq}}}{\text{SFS}_{s^*}} \cdot \left(\mathbf{SL}_{r,p,s^*,d_{last},t,v} + \sum_{I,O} \mathbf{FI}_{r,p,s^*,d_{last},i,t,v,o} \cdot EFF_{r,i,t,v,o} - \sum_{I,O} \mathbf{FO}_{} \right) = \frac{\text{SFS}_{s^{seq}_{next}}}{\text{SFS}_{s^*_{next}}} \cdot \mathbf{SL}_{r,p,s^*_{next},d_{first},t,v} + \tag{10}$$



How sequential seasons chain together for seasonal storage. Hatched area is seasonal_storage_level $SSL_{r,p,s^{seq},t,v}$. Vertical lines are StorageLevel $SL_{r,p,s^*,d,t,v}$. Green line is net seasonal storage level $SSL_{r,p,s^{seq},t,v} + SL_{r,p,s^*,d,t,v}$. Background grey lines show how storage levels from non-sequential seasons are combined in sequential seasons. Dashed line is SeasonalStorageEnergyUpperBound. Sequential seasons two and four here are each two days while one and three are each one day.

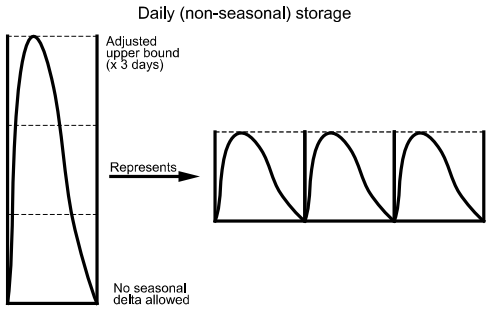
temoa.components.storage.storage_energy_upper_bound_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

This constraint ensures that the amount of energy stored does not exceed the upper bound set by the energy capacity of the storage device, as calculated on the right-hand side.

Because the number and duration of time slices are user-defined, we need to adjust the storage duration, which is specified in hours. First, the hourly duration is divided by the number of hours in a year to obtain the duration as a fraction of the year. Since the $C2A$ parameter assumes the conversion of capacity to annual activity, we need to express the storage duration as fraction of a year. Then, $SEG_{s,d}$ summed over the time-of-day slices (d) multiplied by DPP yields the number of days per season. This step is necessary because conventional time sliced models use a single day to represent many days within a given season. Thus, it is necessary to scale the storage duration to account for the number of days in each season.

$$\mathbf{SL}_{r,p,s,d,t,v} \leq \mathbf{CAP}_{r,p,t,v} \cdot C2A_{r,t} \cdot \frac{SD_{r,t}}{24 \cdot DPP} \cdot \sum_{d} SEG_{s,d} \cdot DPP \quad \forall \{r, p, s, d, t, v\} \in \Theta_{\text{StorageEnergyUpperBound}} \tag{11}$$

A season can represent many days. Within each season, flows are multiplied by the number of days each season represents and, so, the upper bound needs to be adjusted to allow day-scale flows (e.g., charge in the morning, discharge in the afternoon).



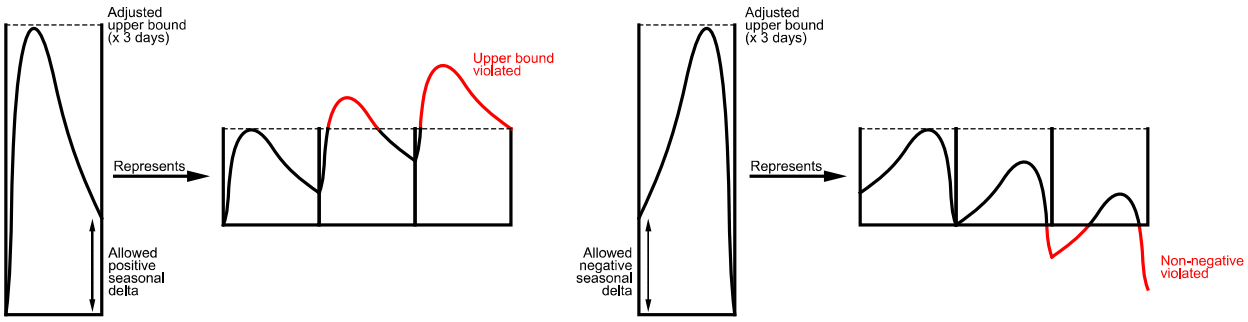
Representation of a 3-day season for non-seasonal (daily) storage.

temoa.components.storage.seasonal_storage_energy_upper_bound_constraint(*model: TemoaModel, r: Region, p: Period, s_seq: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Builds off of StorageEnergyUpperBound_constraint. Enforces the max charge capacity of seasonal storage, summing the real storage level with the superimposed sequential seasonal storage level. s^* represents the matching non-sequential season for the sequential season s^{seq}.

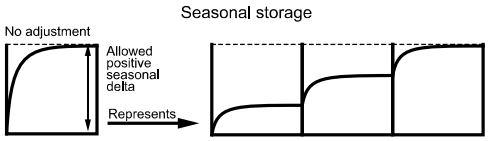
$$\mathbf{SSL}_{r,p,s^{seq},t,v} + \mathbf{SL}_{r,p,s^*,d,t,v} \cdot \frac{\text{SFS}_{s^{seq}}}{\text{SFS}_{s^*}} \leq \mathbf{CAP}_{r,p,t,v} \cdot C2A_{r,t} \cdot \frac{SD_{r,t}}{24 \cdot DPP} \quad (12)$$

Unlike non-seasonal (daily) storage, seasonal storage is allowed to carry energy between seasons. However, through seasons representing multiple days, many days' charge deltas have accumulated, multiplied by the number of days the season represents. If we allowed these stacked deltas to carry between seasons then we would be multiplying the effective energy capacity of the storage. We could just constrain the seasonal delta to the unadjusted energy capacity, but then the final day in the season would sit atop a season's worth of deltas, possibly exceeding our upper or lower bound by a factor of $\frac{N-1}{N}$ where N is the number of days the sequential season represents.



The energy upper bound or non-negative lower bound could be violated in a season representing multiple days if we both adjusted the upper bound to the number of days and allowed a seasonal delta.

So, we do not adjust the upper energy bound for seasonal storage. This limits the ability of seasonal storage to perform arbitrage within each season, but allows it to carry energy between seasons realistically.



Unadjusted energy upper bound constraint for seasonal storage.

temoa.components.storage.storage_charge_rate_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike

This constraint ensures that the charge rate of the storage unit is limited by the power capacity (typically GW) of the storage unit. [source]

$$\sum_{I,O} \mathbf{FIS}_{r,p,s,d,i,t,v,o} \cdot EFF_{r,i,t,v,o} \leq \mathbf{CAP}_{r,p,t,v} \cdot C2A_{r,t} \cdot SEG_{s,d} \quad (13)$$

$$\forall \{r, p, s, d, t, v\} \in \Theta_{\text{StorageChargeRate}}$$

temoa.components.storage.storage_discharge_rate_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike

This constraint ensures that the discharge rate of the storage unit is limited by the power capacity (typically GW) of the storage unit. [source]

$$\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} \leq \mathbf{CAP}_{r,p,t,v} \cdot C2A_{r,t} \cdot SEG_{s,d} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{StorageDischargeRate}} \tag{14}$$

temoa.components.storage.storage_throughput_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike

It is not enough to only limit the charge and discharge rate separately. We also need to ensure that the maximum throughput (charge + discharge) does not exceed the capacity (typically GW) of the storage unit. [source]

$$\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} + \sum_{I,O} \mathbf{FIS}_{r,p,s,d,i,t,v,o} \cdot EFF_{r,i,t,v,o} \leq \mathbf{CAP}_{r,p,t,v} \cdot C2A_{r,t} \cdot SEG_{s,d} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{StorageThroughput}} \tag{15}$$

temoa.components.operations.ramp_up_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

The ramp rate constraint is utilized to limit the rate of electricity generation increase and decrease between two adjacent time slices in order to account for physical limits associated with thermal power plants. This constraint is only applied to technologies in the set `tech_upramping`. We assume for simplicity the rate limits do not vary with technology vintage. The ramp rate limits for a technology should be expressed in percentage of its rated capacity per hour.

In a representative periods or seasonal time slices model, the next time slice, (s_{next}, d_{next}), from the end of each season, (s, d_{last}) is the beginning of the same season, (s, d_{first}). For these time sequencing modes, ramp rates do not apply between seasons.

$$\frac{\sum_{I,O} \mathbf{FO}_{r,p,s_{next},d_{next},i,t,v,o}}{SEG_{s_{next},d_{next}} \cdot 24 \cdot DPP} - \frac{\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o}}{SEG_{s,d} \cdot 24 \cdot DPP} \leq RUH_{r,t} \cdot \Delta H \cdot \frac{CAP_{r,p,t,v} \cdot C2A_{r,t}}{24 \cdot DPP} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{ramp\_up}} \tag{16}$$

$$\text{where: } \Delta H = \frac{H_d + H_{d_{next}}}{2}$$

where:

- $SEG_{s,d}$ is the fraction of the period in time slice (s, d)
- DPP is the number of days in each period
- $RUH_{r,t}$ is the ramp up rate per hour
- ΔH is the average of the hours in timeslice d and d_{next}, i.e. $(H_d + H_{d_{next}})/2$
- $CAP \cdot C2A/(24 \cdot DPP)$ gives the maximum hourly capacity

temoa.components.operations.ramp_down_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage*) → ExprLike [source]

Similar to the `ramp_up` constraint, we use the `ramp_down` constraint to limit ramp down rates between any two adjacent time slices.

$$\frac{\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o}}{SEG_{s,d} \cdot 24 \cdot DPP} - \frac{\sum_{I,O} \mathbf{FO}_{r,p,s_{next},d_{next},i,t,v,o}}{SEG_{s_{next},d_{next}} \cdot 24 \cdot DPP} \leq RDH_{r,t} \cdot \Delta H \cdot \frac{CAP_{r,p,t,v} \cdot C2A_{r,t}}{24 \cdot DPP}$$
$$\forall \{r, p, s, d, t, v\} \in \Theta_{\text{ramp\_down}} \quad (17)$$
$$\text{where: } \Delta H = \frac{H_d + H_{d_{next}}}{2}$$

temoa.components.reserves.planning_reserve_margin_constraint(*model: TemoaModel, r_g: Region, p: Period, s: Season, d: TimeOfDay, t_g: Technology*) → Constraint

During each period p, the sum of capacity values of all reserve technologies, weighted by their planning reserve credit, must exceed the region-group's proxy demand by PRM_{r_g,t_g} in every time slice. [source]

This credit represents the expected availability of nameplate capacity during peak demand conditions and is usually determined from stochastic modelling of demand/supply scenarios. A credit of 1 indicates the technology is expected to be able to output its full nameplate capacity at high demand, while 0 indicates it offers no reliable capacity. Technologies in the tech group are used to calculate the proxy demand and so must include these zero credit processes as well.

The default credit is 0 if not set (i.e., we assume technologies offer no capacity value by default).

For exchange technologies (e.g., inter-regional transmission), reserve contributions are added for capacity into the region-group but *subtracted* for capacity out of it.

(18)

$$
\begin{aligned}
&\sum_{(r,t,v)\in\Theta^{res}\setminus T^x} PRC_{r,t}\cdot \mathbf{CAP}_{r,p,t,v}\cdot SEG_{s,d}\cdot C2A_{r,t} && \text{(production capacity,}\\
&+\sum_{(r,t,v)\in\Theta^{res}\cap T^x} \sigma_{r,r_g}\cdot PRC_{r,t}\cdot \mathbf{CAP}_{r,p,t,v}\cdot SEG_{s,d}\cdot C2A_{r,t} && \text{(net import capacity)}\\
&\geq D^{proxy}_{r_g,p,s,d}\cdot(1+PRM_{r_g,t_g})\\
&\forall\{r_g,p,s,d,t_g\}\in\Theta_{\text{PlanningReserveMargin}}
\end{aligned}
$$

where $\sigma_{r,r_g}=+1$ for an exchange process delivering into region-group r_g and -1 for one delivering out of it, and $D^{proxy}_{r_g,p,s,d}$ is the group's proxy demand defined in (20).

temoa.components.reserves.operating_reserve_margin_constraint(*model: TemoaModel, r_g: Region, p: Period, s: Season, d: TimeOfDay, t_g: Technology*) → Constraint

A dynamic alternative to the planning reserve margin constraint. Capacity values are calculated from process output availability in each time slice, accounting for capacity factors, unit commitment (if the extension is enabled), and a seasonal derating factor which may adjust for, for example, seasonal forced outage rates. A derate factor of 1 indicates no derating while a factor of 0 indicates zero dependable output in that season. Technologies in the tech group are used to calculate the proxy demand and so must include these fully derated processes as well. [source]

The default derate factor is 1 if not set (i.e., we assume technologies are fully available, up to their capacity factor, by default).

For exchange technologies (e.g., inter-regional transmission), reserve contributions are added for available output into the region-group but *subtracted* for capacity out of it.

The availability of storage technologies is a non-trivial problem as it depends on state of charge, which has a temporal dependency (i.e., if we consider a storage technology to contribute reserve in time t can it also contribute in time t+1?). In this implementation, we let storage contribute only what it actually outputs (net) in each time slice, as a conservative but tractable approach (use it or lose it).

$$
\begin{aligned}
&\sum_{(r,t,v)\in\Theta^{res}\setminus T^x\setminus T^s} CFP_{r,s,d,t,v}\cdot ORD_{r,s,t}\cdot \mathbf{CAP}_{r,p,t,v}\cdot SEG_{s,d}\cdot C2A_{r,t} && \text{(firm pr}\\
&+\sum_{(r,t,v)\in\Theta^{res}\cap T^s,\, I,O} (\mathbf{FO}_{r,p,s,d,i,t,v,o}-\mathbf{FI}_{r,p,s,d,i,t,v,o})\cdot ORD_{r,s,t} && \text{(net sto}\\
&+\sum_{(r,t,v)\in\Theta^{res}\cap T^x} \sigma_{r,r_g}\cdot CFP_{r,s,d,t,v}\cdot ORD_{r,s,t}\cdot \mathbf{CAP}_{r,p,t,v}\cdot SEG_{s,d}\cdot C2A_{r,t} && \text{(net firr}\\
&\geq D^{proxy}_{r_g,p,s,d}\cdot(1+ORM_{r_g,t_g})\\
&\forall\{r_g,p,s,d,t_g\}\in\Theta_{\text{OperatingReserveMargin}}
\end{aligned}
\tag{19}
$$

where $\sigma_{r,r_g}=+1$ for an exchange process delivering into region-group r_g and -1 for one delivering out of it, and $D^{proxy}_{r_g,p,s,d}$ is the group's proxy demand defined in (20).

temoa.components.reserves.reserve_margin_proxy_demand(*model: TemoaModel*, *processes: set[tuple[Region, Technology, Vintage]]*, *r_g: Region*, *p: Period*, *s: Season*, *d: TimeOfDay*) → ExprLike [source]

In Temoa, demand for a particular commodity (e.g., electricity) may be endogenous to decisions in the model. So, we estimate demand as the net production of processes in the reserve group, $\Theta^{res}_{r_g,p,t_g}$. This provides the RHS demand for both reserve constraints.

The region group r_g indicates the regions in which demand is met, and the tech group t_g indicates which technologies supply the demanded commodity. The technology group should include all technologies that produce, store, or import/export the commodity of interest but **not** the technologies that demand it downstream.

> **Note**
>
> In Temoa, we are not reasonably able to disaggregate the **available** output of a process for individual commodities when that process outputs multiple commodities. As a result, a reserve margin constraint cannot be applied to a single commodity where supplying processes output multiple different commodities (e.g., if a co-generation plant outputs both heat and electricity, the heat will be included in the equations). This can be avoided by adding an intermediate "dummy" process that throughputs only the output of interest and then adding this dummy process to the tech group instead of the original process, but this requires careful cloning of other technoeconomic parameters so the reserve contributions remain the same.

$$
\begin{aligned}
D^{proxy}_{r_g,p,s,d} = & \sum_{\substack{(r,t,v)\in\Theta^{res}\setminus T^a \\ \uparrow_{r,r_g},\, I,O}} \mathbf{FO}_{r,p,s,d,i,t,v,o} && \text{(no} \\
& + \sum_{\substack{(r,t,v)\in\Theta^{res}\cap T^a \\ \uparrow_{r,r_g},\, I,O}} \begin{cases} DSD_{r,s,d,o} & o \in C^d \\ SEG_{s,d} & \text{otherwise} \end{cases} \cdot \mathbf{FOA}_{r,p,i,t,v,o} && \text{(an} \\
& - \sum_{(r,t,v)\in\Theta^{res}\cap T^s,\, I,O} \mathbf{FI}_{r,p,s,d,i,t,v,o} && \text{(st} \\
& - \sum_{\substack{(r,t,v)\in\Theta^{res}\cap T^x\setminus T^a \\ \downarrow_{r,r_g},\, I,O}} \mathbf{FO}_{r,p,s,d,i,t,v,o}/EFF_{r,p,s,d,i,t,v,o} && \text{(no} \\
& - \sum_{\substack{(r,t,v)\in\Theta^{res}\cap T^x\cap T^a \\ \downarrow_{r,r_g},\, I,O}} \begin{cases} DSD_{r,s,d,o} & o \in C^d \\ SEG_{s,d} & \text{otherwise} \end{cases} \cdot \mathbf{FOA}_{r,p,i,t,v,o}/EFF_{r,p,i,t,v,o} && \text{(an}
\end{aligned}
\tag{20}
$$

where $\uparrow_{r,r_g}$ selects non-exchange processes and exchange imports ($r_2 \in r_g,\ r_1 \notin r_g$), and $\downarrow_{r,r_g}$ selects exchange exports ($r_1 \in r_g,\ r_2 \notin r_g$). $\Theta^{res} = \Theta^{res}_{r_g,p,t_g}$ is the set of all (r,t,v) processes contributing to this reserve margin in this period.

temoa.components.emissions.linked_emissions_tech_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage, e: Commodity*) → ExprLike [source]

This constraint can be used for carbon capture technologies that produce CO2 as an emissions commodity, but the CO2 also serves as a physical input commodity to a downstream process, such as synthetic fuel production. To accomplish this, a dummy technology is linked to the CO2-producing technology, converting the emissions activity into a physical commodity amount as follows:

$$-\sum_{I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} \cdot EAC_{r,e,i,t,v,o} = \sum_{I,O} \mathbf{FO}_{r,p,s,d,i,\hat{t},v,o} \quad \forall \{r,p,s,d,t,v,e\} \in \Theta_{\text{linked\_techs}} \tag{21}$$

where $\hat{t} = LT_{r,t,e}$ is the linked technology for primary technology t and emissions commodity e. For annual technologies ($t \in T^a$ or $\hat{t} \in T^a$), the corresponding $\mathbf{FOA}$ variable scaled by DSD (for end use demand techs) or SEG (for all other annual techs) is used instead.

The relationship between the primary and linked technologies is given in the `linked_techs` table. It is implicit that the primary region corresponds to the linked technology as well. The lifetimes of the primary and linked technologies should be specified and identical.

Objective Function

temoa.components.costs.annuity_to_pv(*rate: float, periods: float*) → float | Expression [source]

Multiplication factor to convert an annuity to net present value

$$\frac{P}{A}(i,N) = \frac{(1+i)^N - 1}{i(1+i)^N} \tag{22}$$

where:

- i is the interest/discount rate
- N is the number of periods

temoa.components.costs.pv_to_annuity(*rate: float, periods: float*) → float | Expression [source]

Multiplication factor to convert net present value to an annuity

$$\frac{A}{P}(i,N) = \frac{i + (1+i)^N}{(1+i)^N - 1} \tag{23}$$

temoa.components.costs.fv_to_pv(*rate: float, periods: float*) → float | Expression

Multiplication factor to convert a future value to net present value [source]

$$\frac{P}{F}(i, N) = \frac{1}{(1+i)^N} \tag{24}$$

temoa.components.costs.total_cost_rule(*model: TemoaModel*) → Expression [source]

Using the `FlowOut` and `Capacity` variables, the Temoa objective function calculates the cost of energy supply, under the assumption that capital costs are paid through loans. This implementation sums up all the costs incurred, and is defined as $C_{tot} = C_{loans} + C_{fixed} + C_{variable} + C_{emissions}$. Each term on the right-hand side represents the cost incurred over the model time horizon and discounted to the initial year in the horizon (P_0). The calculation of each term is given below.

$$\begin{aligned} C_{loans} = & \sum_{r,t,v \in \Theta_{CI}} CI_{r,t,v} \cdot \mathbf{NCAP}_{r,t,v} && \text{(overnight capital cost)} \\ & \cdot \frac{A}{P}(i = \mathrm{LR}_{r,t,v}, N = \mathrm{LLP}_{r,t,v}) && \text{(overnight cost amortised ir} \\ & \cdot \frac{P}{A}(i = GDR, N = \mathrm{LLP}_{r,t,v}) && \text{(annual loan payments disc} \\ & \cdot \frac{A}{P}(i = GDR, N = \mathrm{LTP}_{r,t,v}) && \text{(NPV reamortised over life} \\ & \cdot \frac{P}{A}(i = GDR, N = \min(\mathrm{LTP}_{r,t,v}, P_e - v)) && \text{(costs within planning horiz} \\ & \cdot \frac{P}{F}(i = GDR, N = v - P_0) && \text{(NPV in vintage year disco} \end{aligned} \tag{25}$$

Note that capital costs ($CI_{r,t,v}$) are handled in several steps.

1. Each capital cost is amortized using the loan rate (i.e., technology-specific discount rate) and loan period.
2. The annual stream of payments is converted into a lump sum using the global discount rate and loan period.
3. The new lump sum is amortized at the global discount rate over the process lifetime.
4. Loan payments beyond the model time horizon are removed and the lump sum recalculated.
5. Finally, the lump sum is discounted back to the beginning of the horizon (P_0) using the global discount rate.

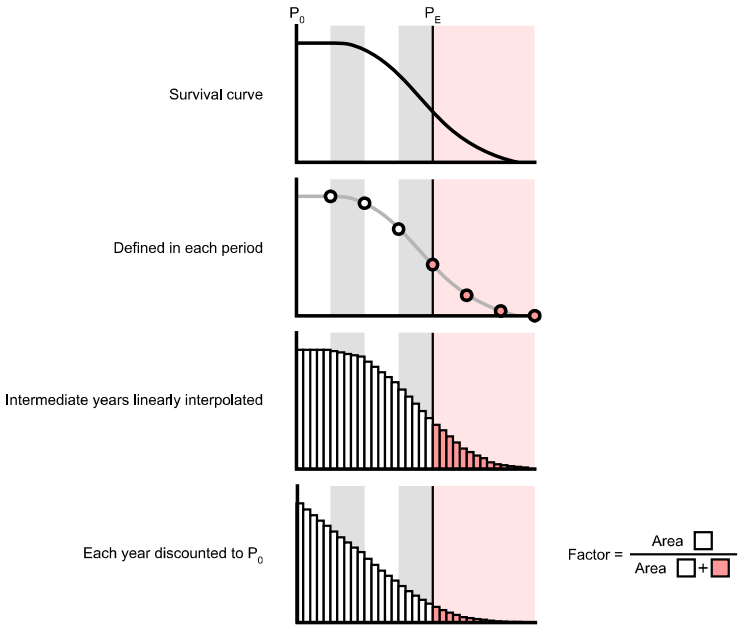
Steps 3 and 4 serve to correctly balance the cost-benefit of technologies whose useful lives would extend beyond the planning horizon. While an explicit salvage term is not included, this approach

properly captures the capital costs incurred within the model time horizon, accounting for process-specific loan rates and periods.

In the case of processes using survival curves, steps 3 and 4 do not reamortise costs uniformly over the process lifetime. Instead, costs are amortised over the life of the process in proportion to the survival fraction in each year. Note that, for this calculation, a survival curve $LSC_{r,y,t,v}$ must be defined out to the year in which the surviving fraction is zero, even if that extends beyond the planning horizon. It must also be defined for each integer year between model periods and, if not, these gaps will be filled by linear interpolation ahead of this calculation.

$$
\begin{aligned}
C_{loans,LSC} = & \sum_{r,t,v \in \Theta_{CI}} CI_{r,t,v} \cdot \mathbf{NCAP}_{r,t,v} && \text{(overnight cap} \\
& \cdot \frac{A}{P}(i = \mathrm{LR}_{r,t,v}, N = \mathrm{LLP}_{r,t,v}) && \text{(overnight cos} \\
& \cdot \frac{P}{A}(i = GDR, N = \mathrm{LLP}_{r,t,v}) && \text{(annual loan p} \\
& \cdot \left(\sum_{v<Y} LSC_{r,y,t,v} \cdot \frac{P}{F}(i = GDR, N = P - v + 1) \right)^{-1} && \text{(reamortised c} \\
& \cdot \sum_{v<Y<P_e} LSC_{r,y,t,v} \cdot \frac{P}{F}(i = GDR, N = P - v + 1) && \text{(costs within p} \\
& \cdot \frac{P}{F}(i = GDR, N = v - P_0) && \text{(NPV in vinta}
\end{aligned}
\tag{26}
$$

Where Y is the set of each integer year y within the planning horizon.



Steps 3 and 4 for processes with survival curves.

Fixed, variable, and emissions annual cost factors are determined by:

$$
\begin{aligned}
C_{fixed} &= \sum_{r,p,t,v \in \Theta_{CF}} CF_{r,p,t,v} \cdot \mathbf{CAP}_{r,p,t,v} && \text{(annual fixed cos} \\
C_{variable} &= \sum_{r,p,t \notin T^a, v \in \Theta_{CV}} CV_{r,p,t,v} \cdot \sum_{S,D,I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} && \text{(annual variable} \\
&\quad \text{where } t \notin T^a \\
&\quad + \\
&\quad \sum_{r,p,t \in T^a,\ v \in \Theta_{VC}} CV_{r,p,t,v} \cdot \sum_{I,O} \mathbf{FOA}_{r,p,i,t,v,o} && \text{(annual variable} \\
&\quad \text{where } t \in T^a \\
&\quad + \\
C_{emissions} &= \sum_{r,p,e \in \Theta_{CE}} CE_{r,p,e} \cdot EAC_{r,e,i,t,v,o} \cdot \sum_{S,D,I,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} && \text{(annual emission} \\
&\quad \text{where } t \notin T^a \\
&\quad + \\
&\quad \sum_{r,p,e \in \Theta_{CE}} CE_{r,p,e} \cdot EAC_{r,e,i,t,v,o} \cdot \sum_{I,O} \mathbf{FOA}_{r,p,i,t,v,o} && \text{(annual emission} \\
&\quad \text{where } t \in T^a \\
&\quad + \\
&\quad \sum_{r,p,e \in \Theta_{CE}} \frac{CE_{r,p,e} \cdot EE_{r,e,t,v} \cdot \mathbf{NCAP}_{r,t,v=p}}{LEN_p} && \text{(annual embodied} \\
&\quad + \\
&\quad \sum_{r,p,e \in \Theta_{CE},v} CE_{r,p,e} \cdot EEOL_{r,e,t,v} \cdot \mathbf{ART}_{r,p,t,v} && \text{(annual retiremer}
\end{aligned}
\tag{27}
$$

Each of these costs are then discounted within each period and then to the base year:

$$
\begin{aligned}
C_{fix,var,emiss} =& C_{fixed} + C_{variable} + C_{emissions} \\
& \cdot \frac{P}{A}(i = GDR,\ N = LEN_p) && \text{(for each year in period } p \text{ discount} \\
& \cdot \frac{P}{F}(i = GDR,\ N = p - P_0) && \text{(discounted from period } p \text{ to NPV}
\end{aligned}
\tag{28}
$$

User-Specific Constraints

temoa.components.limits.limit_emission_constraint(*model: TemoaModel, r: Region, p: Period, e: Commodity, op: str*) → ExprLike [source]

A modeler can track emissions through use of the `commodity_emissions` set and `emission_activity` parameter. The EAC parameter is analogous to the efficiency table, tying emissions to a unit of activity. The limit_emission constraint allows the modeler to assign an upper bound per period to each emission commodity. Note that this constraint sums emissions from technologies with output varying at the time slice and those with constant annual output in separate terms. It also includes embodied emissions from new capacity and end-of-life emissions from retiring capacity.

$$\begin{aligned}
&\sum_{S,D,I,T,V,O|r,e,i,t,v,o\in EAC} (EAC_{r,e,i,t,v,o} \cdot \mathbf{FO}_{r,p,s,d,i,t,v,o}) \\
&+ \sum_{I,T,V,O|r,e,i,t\in T^a,v,o\in EAC} (EAC_{r,e,i,t,v,o} \cdot \mathbf{FOA}_{r,p,i,t\in T^a,v,o}) \\
&+ \sum_{T} \frac{EE_{r,e,t,v=p} \cdot \mathbf{NCAP}_{r,t,v=p}}{LEN_p} \\
&+ \sum_{T,V} EEOL_{r,e,t,v} \cdot \mathbf{ART}_{r,p,t,v} \\
&\leq, \geq, \text{or} = \quad LE_{r,p,e} \\
&\forall \{r,p,e\} \in \Theta_{\text{limit_emission}}
\end{aligned} \tag{29}$$

temoa.components.limits.limit_activity_constraint(*model: TemoaModel*, *r: Region*, *p: Period*, *t: Technology*, *op: str*) → ExprLike [source]

Sets a limit on the activity from a specific technology. Note that the indices for these constraints are region, period and tech, not tech and vintage. The first version of the constraint pertains to technologies with variable output at the time slice level, and the second version pertains to technologies with constant annual output belonging to the `tech_annual` set.

$$\begin{aligned}
&\sum_{S,D,I,V,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} \\
&\forall \{r,p,t \notin T^a\} \in \Theta_{\text{limit_activity}} \\
&+ \sum_{I,V,O} \mathbf{FOA}_{r,p,i,t\in T^a,v,o} \\
&\forall \{r,p,t \in T^a\} \in \Theta_{\text{limit_activity}} \\
&\leq, \geq, \text{or} = \quad LA_{r,p,t}
\end{aligned} \tag{30}$$

temoa.components.limits.limit_activity_share_constraint(*model: TemoaModel*, *r: Region*, *p: Period*, *g1: Technology*, *g2: Technology*, *op: str*) → ExprLike [source]

Limits the activity of a given technology or group as a fraction of another technology or group, summed over a period. This can be used to set, for example, a renewable portfolio scheme constraint.

$$\begin{aligned}
&\sum_{R_g\subseteq R,\ S,\ D,\ I,\ (T^{g_1}\setminus T^a)\subseteq T,\ V,\ O} \mathbf{FO}_{r,p,s,d,i,t,v,o} + \sum_{R_g\subseteq R,\ I,\ (T^{g_1}\cap T^a)\subseteq T,\ V,\ O} \mathbf{FOA}_{r,p} \\
&\leq, \geq, \text{or} \\
&LAS_{r,p,g_1,g_2} \cdot \sum_{R_g\subseteq R,\ S,\ D,\ I,\ (T^{g_2}\setminus T^a)\subseteq T,\ V,\ O} \mathbf{FO}_{r,p,s,d,i,t,v,o} + \sum_{R_g\subseteq R,\ I,\ (T^{g_2}\cap T^a)\subseteq T,\ V,\ O} \mathbf{FOA}_{r,p} \\
&\forall \{r,p,g_1,g_2\} \in \Theta_{\text{limit_activity}}
\end{aligned} \tag{31}$$

temoa.components.limits.limit_capacity_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, op: str*) → ExprLike [source]

The limit_capacity constraint sets a limit on the available capacity of a given technology. Note that the indices for these constraints are region, period and tech, not tech and vintage.

$$\begin{aligned}\mathbf{CAPAVL}_{r,p,t} \quad &\leq, \geq, \text{or} = \quad LC_{r,p,t} \\ &\forall \{r,p,t\} \in \Theta_{\text{limit\_capacity}}\end{aligned} \tag{32}$$

temoa.components.limits.limit_new_capacity_constraint(*model: TemoaModel, r: Region, t: Technology, v: Vintage, op: str*) → ExprLike [source]

The limit_new_capacity constraint sets a limit on the newly installed capacity of a given technology or group in a given vintage year.

$$\mathbf{NCAP}_{r,t,v} \quad \leq, \geq, \text{or} = \quad LNC_{r,t,v} \tag{33}$$

temoa.components.limits.limit_capacity_share_constraint(*model: TemoaModel, r: Region, p: Period, g1: Technology, g2: Technology, op: str*) → ExprLike [source]

The limit_capacity_share constraint limits the available capacity of a given technology or technology group as a fraction of another technology or group.

temoa.components.limits.limit_new_capacity_share_constraint(*model: TemoaModel, r: Region, g1: Technology, g2: Technology, v: Vintage, op: str*) → ExprLike [source]

The limit_new_capacity_share constraint limits the share of new capacity of a given technology or group as a fraction of another technology or group.

temoa.components.limits.limit_resource_constraint(*model: TemoaModel, r: Region, t: Technology, op: str*) → ExprLike [source]

The limit_resource constraint sets a limit on the available resource of a given technology across all model time periods. Note that the indices for these constraints are region and tech.

$$\begin{aligned}&\sum_{P,S,D,I,V,O} \mathbf{FO}_{r,p,s,d,i,t \notin T^a,v,o} \\ &+ \sum_{P,I,V,O} \mathbf{FOA}_{r,p,i,t \in T^a,v,o} \\ &\leq, \geq, \text{or} = \quad LS_{r,t} \\ &\forall \{r,t\} \in \Theta_{\text{limit\_resource}}\end{aligned} \tag{34}$$

temoa.components.limits.limit_tech_input_split_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, i: Commodity, t: Technology, v: Vintage, op: str*) → ExprLike [source]

Allows users to limit shares of commodity inputs to a process producing a single output. These shares can vary by model time period. See limit_tech_output_split_constraint for an analogous explanation. Under this constraint, only the technologies with variable output at the timeslice level (i.e., NOT in the `tech_annual` set) are considered.

temoa.components.limits.limit_tech_output_split_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage, o: Commodity, op: str*) → ExprLike [source]

Some processes take a single input and make multiple outputs, and the user would like to specify either a constant or time-varying ratio of outputs per unit input. The most canonical example is an oil refinery. Crude oil is used to produce many different refined products. In many cases, the modeler would like to limit the share of each refined product produced by the refinery.

For example, a hypothetical (and highly simplified) refinery might have a crude oil input that produces 4 parts diesel, 3 parts gasoline, and 2 parts kerosene. The relative ratios to the output then are:

$$d = \frac{4}{9} \cdot \text{total output}, \qquad g = \frac{3}{9} \cdot \text{total output}, \qquad k = \frac{2}{9} \cdot \text{total output}$$

Note that it is possible to specify output shares that sum to less than unity. In such cases, the model optimizes the remaining share. In addition, it is possible to change the specified shares by model time period. Under this constraint, only the technologies with variable output at the timeslice level (i.e., NOT in the `tech_annual` set) are considered.

The constraint is formulated as follows:

$$\sum_{I,t \notin T^a} \mathbf{FO}_{r,p,s,d,i,t,v,o} \quad \leq, \geq, \text{or} = \quad TOS_{r,p,t,o} \cdot \sum_{I,O,t \notin T^a} \mathbf{FO}_{r,p,s,d,i,t,v,o} \qquad \forall \{r,p,s,d,t,v,o\} \in \Theta_{\text{limit\_tech\_output\_split}} \tag{35}$$

temoa.components.limits.limit_tech_input_split_annual_constraint(*model: TemoaModel, r: Region, p: Period, i: Commodity, t: Technology, v: Vintage, op: str*) → ExprLike [source]

Allows users to limit shares of commodity inputs to a process producing a single output. These shares can vary by model time period. See limit_tech_output_split_annual_constraint for an analogous explanation. Under this function, only the technologies with constant annual output (i.e., members of the `tech_annual` set) are considered.

temoa.components.limits.limit_tech_output_split_annual_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage, o: Commodity, op: str*)

→ **ExprLike** [source]

This constraint operates similarly to limit_tech_output_split_constraint. However, under this function, only the technologies with constant annual output (i.e., members of the `tech_annual` set) are considered.

$$\sum_{I,T^a} \mathbf{FOA}_{r,p,i,t\in T^a,v,o} \quad \leq, \geq, \text{or} = \quad TOSA_{r,p,t,o} \cdot \sum_{I,O,T^a} \mathbf{FOA}_{r,p,i,t\in T^a,v,o} \qquad \forall \{r,p,t\in T^a,v,o\} \in \Theta_{\text{limit\_tech\_output\_split\_annual}} \tag{36}$$

temoa.components.limits.limit_tech_input_split_average_constraint(*model: TemoaModel, r: Region, p: Period, i: Commodity, t: Technology, v: Vintage, op: str*)

→ **ExprLike** [source]

Allows users to limit shares of commodity inputs to a process producing a single output. Under this constraint, only the technologies with variable output at the timeslice level (i.e., NOT in the `tech_annual` set) are considered. This constraint differs from limit_tech_input_split as it specifies shares on an annual basis, so even though it applies to technologies with variable output at the timeslice level, the constraint only fixes the input shares over the course of a year.

temoa.components.limits.limit_tech_output_split_average_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage, o: Commodity, op: str*)

→ **ExprLike** [source]

Allows users to limit shares of commodity outputs from a process. Under this constraint, only the technologies with variable output at the timeslice level (i.e., NOT in the `tech_annual` set) are considered. This constraint differs from limit_tech_output_split as it specifies shares on an annual basis, so even though it applies to technologies with variable output at the timeslice level, the constraint only fixes the output shares over the course of a year.

temoa.components.limits.limit_annual_capacity_factor_constraint(*model: TemoaModel, r: Region, p: Period, t: Technology, v: Vintage, o: Commodity, op: str*)

→ **ExprLike** [source]

The limit_annual_capacity_factor sets an upper bound on the annual capacity factor from a specific process. The first portion of the constraint pertains to technologies with variable output at the time slice level, and the second portion pertains to technologies with constant annual output belonging to the `tech_annual` set.

$$\sum_{S,D,I} \mathbf{FO}_{r,p,s,d,i,t,v,o} \quad \leq, \geq, \text{or} = \quad LIMACF_{r,t,v,o} \cdot \mathbf{CAP}_{r,p,t,v} \cdot \text{C2A}_{r,t} \quad \forall \{r,p,t \notin T^a, v, o\} \in \Theta_{\text{limit\_annual\_capacity\_factor}}$$

$$\sum_{I} \mathbf{FOA}_{r,p,i,t,v,o} \quad \leq, \geq, \text{or} = \quad LIMACF_{r,t,v,o} \cdot \mathbf{CAP}_{r,p,t,v} \cdot \text{C2A}_{r,t} \quad \forall \{r,p,t \in T^a, v, o\} \in \Theta_{\text{limit\_annual\_capacity\_factor}} \tag{37}$$

temoa.components.limits.limit_seasonal_capacity_factor_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, t: Technology, op: str*) → ExprLike

The limit_seasonal_capacity_factor sets an upper bound on the seasonal capacity factor from a specific technology. The first portion of the constraint pertains to technologies with variable output at the time slice level, and the second portion pertains to technologies with constant annual output belonging to the `tech_annual` set. [source]

$$\sum_{D,I,V,O} \mathbf{FO}_{r,p,s,d,i,t,v,o} \quad \leq, \geq, \text{or} = \quad LIMSCF_{r,s,t} \cdot \mathbf{CAPAVL}_{r,p,t} \cdot \text{C2A}_{r,t} \cdot SF \quad \forall \{r,p,s,t \notin T^a\} \in \Theta_{\text{limit\_seasonal\_capacity\_fact}}$$

$$\sum_{I,V,O} \mathbf{FOA}_{r,p,i,t,v,o} \cdot SFS_s \quad \leq, \geq, \text{or} = \quad LIMSCF_{r,s,t} \cdot \mathbf{CAPAVL}_{r,p,t} \cdot \text{C2A}_{r,t} \cdot SF \quad \forall \{r,p,s,t \in T^a\} \in \Theta_{\text{limit\_seasonal\_capacity\_fact}} \tag{38}$$

temoa.components.storage.limit_storage_fraction_constraint(*model: TemoaModel, r: Region, p: Period, s: Season, d: TimeOfDay, t: Technology, v: Vintage, op: str*) → ExprLike [source]

This constraint is used if the users wishes to force a specific storage charge level for certain storage technologies and vintages at a certain time slice. User-specified storage charge levels that are sufficiently different from the optimal could impact the cost-effectiveness of storage.

`s` can be a season from the `time_season` set or a sequential season from the `time_sequential_season` set.

$$\frac{\mathbf{SL}_{r,p,s,d,t,v}}{\text{SFS}_s \cdot \text{DPP}} \quad \leq, \geq, \text{or} = \quad \mathbf{CAP}_{r,p,t,v} \cdot \text{C2A}_{r,t} \cdot \frac{\text{SD}_{r,t}}{24 \cdot \text{DPP}} \cdot \text{LSF}_{r,s,d,t} \quad \forall \{r,p,s,d,t,v\} \in \Theta_{\text{limit\_storage\_fraction}} \tag{39}$$

For seasonal storage technologies, the LHS becomes:

$$\mathbf{SSL}_{r,p,s_{seq},t,v} + \frac{\mathrm{SFS}_{s_{seq}}}{\mathrm{SFS}_{s^*}} \cdot \mathbf{SL}_{r,p,s^*,d,t,v}$$

General Caveats

Temoa does not currently provide an easy avenue to track multiple concurrent energy flows through a process. Consider a cogeneration plant. Where a conventional power plant might simply emit excess heat as exhaust, a cogeneration plant harnesses some or all of that heat for heating purposes, either very close to the plant, or generally as hot water for district heating. Temoa's flow variables can track both flows through a process, but each flow will have its own efficiency from the efficiency parameter. This implies that to produce 1 unit of electricity will require $\frac{1}{elceff}$ units of input. At the same time, to produce 1 unit of heat will require units of input energy, and to produce both output units of heat and energy, both flows must be active, and the desired activity will be double-counted by Temoa.

To model a parallel output device (c.f., a cogeneration plant), the modeler must currently set up the process with the `tech_input_split` and `tech_output_split` parameters, appropriately adding each flow to the efficiency parameter and accounting for the overall process efficiency through all flows.