**Optimization Problem Formulation (Python E-hub Model, Dev v5.4)**

# Objective

Minimize total cost ()

Or

Minimize total carbon ()

# Sets

| **Set** | **Python definition** | **Description** | **Excel input / defining criteria** |
| --- | --- | --- | --- |
|  | model.Time | Set of time periods | *Demand data* sheet, number of rows (hours) |
|  | model.SubTime | Set of time periods including time 0  (used in state-of-charge constraint where time-1 is needed) |  |
|  | model.Tech | Set of technologies | *Energy Converters* sheet, number of technology columns (T); technology IDs assigned from 1 to T |
|  | model.Stg | Set of storage technologies | *Storage* sheet, number of storage technology columns (S); storage IDs assigned from 1 to S |
|  | model.LinkID | Set of network link IDs | *Network* sheet, number of connection columns (L); link IDs assigned from 1 to L |
|  | model.EC | Set of energy carrier demands | *Energy Carriers* sheet, number of energy carrier rows (E); EC IDs assigned from 1 to E |
|  | model.SolEC | Solar energy carrier ID | *Solar Profile* sheet, *Solar energy carrier* field |
|  | model.SolarTechs | Set of solar technologies | *Energy Converters* sheet, technologies with a solar energy carrier input |
|  | model.roof\_tech | Set of rooftop technologies | *Energy Converters* sheet, technologies with a specified *Solar specific power (kW/m2)* (custom constraint) |
|  | model.TechFreeOut | Set of technologies without fixed output shares | *Energy Converters* sheet, technologies with no *Fixed output share* specified (or a single output share of 1 specified) |
|  | model.TechFixOut | Set of technologies with fixed output shares | *Energy Converters* sheet, technologies with more than one (comma-delimited) *Fixed output share* specified |
|  | model.TechFixIn | Set of technologies with fixed input shares | *Energy Converters* sheet, technologies with more than one (comma-delimited) *Fixed input share* specified |
|  | model.PartLTech | Set of part load technologies | *Energy Converters* sheet, technologies with *Minimum load (%)* specified |
|  | model.PartLFree | Set of part load technologies without fixed output shares | *Energy Converters* sheet, partload technologies with no *Fixed output share* specified (or a single output share of 1 specified) |
|  | model.PartLFix | Set of part load technologies with fixed output shares | *Energy Converters* sheet, partload technologies with more than one (comma-delimited) *Fixed input share* specified |
|  | model.EImpSet | Set of import energy carriers | *Imports* sheet, energy carriers listed |
|  | model.EExpSet | Set of export energy carriers | *Exports* sheet, energy carriers listed |
|  | model.ENImpSet | Set of energy carriers which cannot be imported | Energy carriers not listed in *Imports* sheet |
|  | model.ENExpSet | Set of energy carriers which cannot be exported | Energy carriers not listed in *Exports* sheet |
|  | model.hubs | Set of hubs | *General* sheet, *Number of hubs* |
|  | model.hub\_i | Set of hubs i (used to connect hub *hi* to hub *hj*) | Same as *h* |
|  | model.hub\_j | Set of hubs j (used to connect hub *hi* to hub *hj*) | Same as *h* |

# Parameters

| **Parameter** | **Python definition** | **Description** | **Excel input / definition criteria** |
| --- | --- | --- | --- |
|  | model.eff | Efficiency of technology *tech*; applies to technologies with no fixed output share specified (technologies *tfro*) (%) | *Energy Converters* sheet, *Efficiency (%)* |
|  | model.effFixOut | Efficiency of technology *tfxo* for energy carrier *ec* (%) | *Energy Converters* sheet, *Efficiency (%)* and *Fixed output share;*  Where corresponds to the first-listed fixed output share for technology *tfxo* |
|  | model.shrFixIn | Fixed input share for technology *tfxi* for input energy carrier *ec* (fraction) | *Energy Converters* sheet, *Fixed input share*, comma-delimited values which must sum to one; values provided in the same order as ECs listed under *Input energy carrier* |
|  | model.partLoad | Minimum loading of technology *tech* (%) | *Energy Converters* sheet, *Minimum load (%)*; applies to first-listed output EC for multi-output techs with fixed output shares. Otherwise, minimum load applies to total output. |
|  | model.lifeTechs | Lifetime of technology *tech* (years) | *Energy Converters* sheet*, Lifetime (years)* |
|  | model.maxCapTechs | Maximum installed capacity of technology *tech* in hub *h* (kW) | *Energy Converters* sheet, *Maximum capacity (kW)*; set to zero where technology *tech* is not present in hub *h*; set to *infinity* if not provided |
|  | model.minCapTechs | Minimum installed capacity of technology *tech* in hub *h* (kW) | *Energy Converters* sheet, *Minimum* *capacity (kW)*; set to zero where technology *tech* is not present in hub *h* |
|  | model.maxOutTechs | Maximum energy output of technology *tech* in hub *h* (kWh) | *Energy Converters* sheet, *Maximum output (kWh);*  applies to first-listed output EC for multi-output techs with fixed output shares. Otherwise, applies to total output. Set to zero where technology *tech* is not present in hub *h* |
|  | model.minOutTechs | Minimum energy output of technology *tech* in hub *h* (kWh) | *Energy Converters* sheet, *Minimum output (kWh);*  applies to first-listed output EC for multi-output techs with fixed output shares. Otherwise, applies to total output. Set to zero where technology *tech* is not present in hub *h* |
|  | model.techCO2 | Carbon factor associated with the installation of technology *tech* (kg-CO2/kW) | *Energy Converters* sheet, *CO2 investment (kg-CO2/kW)* |
|  | model.YtCapCost | Binary indicator to determine if investment cost should be considered for technology *tech* (i.e., investment cost is ignored if technology is already installed) | *Energy Converters* sheet; set to 1 if capacity is not specified (i.e., not installed); set to 0 if capacity is specified |
|  | model.solkWm2 | Specific power of solar panel technology *tech* (kW/m2) | *Energy Converters* sheet; *Solar specific power (kW/m2)* |
|  |  |  |  |
|  | model.CRFtech | Capital recovery factor for technology *tech* | where i is the interest rate and N is the lifetime of technology *tech* |
|  | model.invTech | Investment cost of technology *tech* (CHF/kW) | *Energy Converters* sheet, *Investment cost (CHF/kW)* |
|  | model.omvTech | Variable operation and maintenance cost for technology *tech* (CHF/kWh) | *Energy Converters* sheet, *Variable O&M cost (CHF/kWh)* |
|  | model.omfTech | Fixed operation and maintenance cost for technology *tech* (CHF/kW) | *Energy Converters* sheet, *Fixed O&M cost (CHF/kW)* |
|  | model.chargingEff | Charging efficiency of storage *stg* (%) | *Storage* sheet, *Charging efficiency (%)* |
|  | model.dischargingEff | Discharging efficiency of storage *stg* (%) | *Storage* sheet, *Discharging efficiency (%)* |
|  | model.maxStorCh | Maximum charging rate for storage *stg* (%) | *Storage* sheet, *Maximum charging rate (%)* |
|  | model.maxStorDisch | Maximum discharging rate for storage *stg* (%) | *Storage* sheet, *Maximum discharging rate (%)* |
|  | model.standbyLoss | Standby loss for storage *stg* (%/hour) | *Storage* sheet, *Standby loss (%/hour)* |
|  | model.minSoC | Minimum state-of-charge for storage *stg* as a share of the total capacity (%) | *Storage* sheet, *Minimum SoC (%)* |
|  | model.invStg | Investment cost for storage *stg* (CHF/kWh) | *Storage* sheet, *Investment cost (CHF/kWh)* |
|  | model.lifeStg | Lifetime of storage *stg* (years) | *Storage* sheet, *Lifetime (years)* |
|  | model.omfStg | Fixed operation and maintenance cost for storage *stg* (CHF/kWh) | *Storage* sheet, *Fixed O&M cost (CHF/kWh)* |
|  | model.CRFstg | Capital recovery factor for storage *stg* | where i is the interest rate and N is the lifetime of storage *stg* |
|  | model.stgCO2 | Carbon factor associated with the installation of storage stg (kg-CO2/kWh) | *Storage* sheet, *CO2 investment (kg-CO2/kWh)* |
|  | model.maxCapStg | Maximum installed capacity of storage *stg* in hub *h* for energy carrier *ec* (kWh) | *Storage* sheet, *Maximum capacity (kWh)*; set to zero where storage *stg* is not present in hub *h* or does not correspond to energy carrier *ec*; set to *infinity* if not provided |
|  | model.minCapStg | Minimum installed capacity of storage *stg* in hub *h* for energy carrier *ec* (kWh) | *Storage* sheet, *Maximum capacity (kWh)*; set to zero where storage *stg* is not present in hub *h* or does not correspond to energy carrier *ec* |
|  | model.YsCapCost | Binary indicator to determine if investment cost should be considered for storage *stg* (i.e., investment cost is ignored if storage is already installed) | *Storage* sheet; set to 1 if capacity is not specified (i.e., not installed); set to 0 if capacity is specified |
|  | model.netLength | Length (m) of network link *lk* | *Network* sheet, *Length (m)* |
|  | model.netLoss | Transport loss (i.e., efficiency) of network link *lk* (fraction/m) | *Network* sheet, *Network loss (fraction/m);* total loss for network link *lk* = netLoss \* netLength |
|  | model.invNet | Investment cost for network link *lk* (CHF/kW/m) | *Network* sheet, *Investment cost (CHF/kW/m)* |
|  | model.omfNet | Fixed operation and maintenance cost for network link *lk* (CHF/kW) | *Network* sheet, *Fixed O&M cost (CHF/kW)* |
|  | model.omvNet | Variable operation and maintenance cost for network link *lk* (CHF/kWh) | *Network* sheet, *Variable O&M cost (CHF/kWh)* |
|  | model.CRFnet | Capital recovery factor for network link *lk* | where i is the interest rate and N is the lifetime of link *lk* |
|  | model.lifeNet | Lifetime of network link *lk*  (years) | *Network* sheet, *Lifetime (years)* |
|  | model.netMax | Maximum installed capacity of network link *lk* between hub *hi* and *hj*, for energy carrier *ec* (kW) | *Network* sheet, *Maximum capacity (kW)*; set to zero for configurations that are not valid (e.g., network link *lk* not present for hub *hi*); set to *infinity* if not provided |
|  | model.netMin | Minimum installed capacity of network link *lk* between hub *hi* and *hj*, for energy carrier *ec* (kW) | *Network* sheet, *Minimum capacity (kW)*; set to zero for configurations that are not valid (e.g., network link *lk* not present for hub *hi*) |
|  | model.netCO2 | Carbon factor associated with the installation of network link *lk* (kg-CO2/kW/m) | *Network* sheet, *CO2 investment (kg-CO2/kW/m)* |
|  | model.Yx | Binary indicator for the existence of a connection between hub *hi* and *hj* for link *lk* and energy carrier *ec* | Set to 1 if connection configuration exists; set to 0 if it does not |
|  | model.YxCapCost | Binary indicator to determine if investment cost should be considered for link *lk* between hub *hi* and *hj* for energy carrier *ec* (i.e., investment cost is ignored if network link is already installed, or if cost has already been accounted for from *hj* to *hi*) | *Network* sheet; set to 1 if capacity is not specified (i.e., not installed); set to 0 if capacity is specified or if cost has already been accounted for from *hj* to *hi* |
|  | model.impCost | Import energy carrier cost (CHF/kWh) | *Imports* sheet, *Price (CHF/kWh)* |
|  | model.expPrice | Export energy carrier price (CHF/kWh) | *Exports* sheet, *Export Price (CHF/kWh)* |
|  | model.co2Tax | Carbon tax associated with the usage of energy carrier *eimp* (CHF/kg-CO2) | *Imports* sheet, *CO2 tax (CHF/kg-CO2)* |
|  | model.maxImp | Maximum supply of import energy carrier *eimp* (kWh) | *Imports* sheet, *Maximum supply (kWh)* |
|  | model.ecCO2 | Carbon factor associated with the use (e.g., combustion) of energy carrier *eimp* (kg CO2/kWh) | *Imports* sheet, *CO2 (kg-CO2/kWh)* |
|  | model.interestRate | Interest rate (%) | *General* sheet, *Interest rate (%)* |
|  | model.solarEm | Insolation in time period *tm* (kW/m2) | *Solar Profile* sheet, *Irradiation (kW/m2)* |
|  | model.maxCarbon | MaximumtotalCO2 emissions across all hubs (kg CO2) | *General* sheet, *Maximum CO2 emissions (kg-CO2)* |
|  | model.bigM | Used as a bound in constraints. Should exceed possible part load technology energy production and network energy exchanges. (BigM acts as an upper limit for Eout from part load technologies and network link exchanges.) | *General* sheet, *Big M* |
|  | model.loads | Demand for energy carrier *ec* in hub *h* during time period *tm* | *Demand* sheet |
|  | model.maxSolarArea | Maximum solar area (m2) | Initialized to 500; (custom constraint) |

# Variables

| **Variable** | **Python definition** | **Description** |
| --- | --- | --- |
| ***Technology variables*** | | |
|  | model.Ein | Input energy *ec* to technology *tech* in hub *h* during time period *tm* (kWh) |
|  | model.Eout | Output energy *ec* from technology *tech* in hub *h* during time period *tm* (kWh) |
|  | model.Eexp | Export energy carrier *ec* from hub *h* during time period *tm* (kWh) |
|  | model.Eimp | Import energy carrier *ec* from hub *h* during time period *tm* (kWh) |
|  | model.CapTech | Installed capacity for technology *tech* in hub *h* (kW) |
|  | model.Ypl\_op | Binary indicator for the operation of partload technology *plt* in hub *h* during time period *tm*.  Note: If Eout from plt > 0, YNon== 1; under other conditions, indicator has no meaning |
| ***Network variables*** | | |
|  | model.CapNet | Network capacity for link *lk* between hub *hi* and *hj* for energy carrier *ec* (kW) |
|  | model.NetE | Energy carrier *ec* exchanged from hub *hi* to *hj* for link *lk* during time period *tm* (kWh) |
|  | model.Yx\_op | Binary indicator for the exchange of energy carrier *ec* from hub *hi* to *hj* for link *lk* during time period *tm*  Note: If NetE > 0, YNx\_op == 1; under other conditions, indicator has no meaning |
| ***Storage variables*** | | |
|  | model.InStg | Energy carrier *ec* stored in technology *stg* in hub *h* during time period *tm* (i.e., charging) (kWh) |
|  | model.OutStg | Energy carrier *ec* discharged from technology *stg* in hub *h* during time period *tm* (i.e., charging) (kWh) |
|  | model.SoC | State-of-charge; i.e., stored energy *ec* in technology *stg* in hub *h* during time period *tm* (kWh) |
|  | model.CapStg | Storage capacity of technology *stg*, which stores energy carrier *ec*,in hub *h* (kWh) |
|  | model.YstgIn | Binary indicator for the storage (charging) of energy carrier *ec* into storage *stg* in hub *h* during time period *tm* |
|  | model.YstgOut | Binary indicator for the discharging of energy carrier *ec* from storage *stg* in hub *h* during time period *tm* |
| ***Totals variables*** | | |
|  | model.TotalCost | Total cost (CHF) |
|  | model.FuelCost | Fuel cost (CHF) |
|  | model.VOMCost | Variable operation and maintenance cost (CHF) |
|  | model.FOMCost | Fixed operation and maintenance cost (CHF) |
|  | model.CO2Tax | Carbon tax (CHF) |
|  | model.IncomeExp | Income from electricity exports (CHF) |
|  | model.InvCost | Investment cost (CHF) |
|  | model.TotalCarbon | Total carbon emissions (kg CO2) |
|  | model.TotalCarbon2 | Total carbon emissions variable needed for Pyomo implementation (kg CO2) |
|  | model.FuelCO2 | Carbon emissions due to energy carrier usage |
|  | model.TechCO2 | Carbon emissions due to technology installation |
|  | model.StgCO2 | Carbon emissions due to storage installation |
|  | model.NetCO2 | Carbon emissions due to network installation |

# Constraints

## Energy balance constraint

Python code reference: *energyBalance\_rule*

i.e., For each hub, time period, and energy carrier:

Demand + exports = imports + storage output – storage input + energy output from technologies – energy input to technologies + network imports – network exports

Energy input = energy output

## Import/export constraints

Maximum import bound

Python code reference: *maxEimp\_rule*

i.e., For each import energy carrier

Total import <= maximum allowable import

Export/import setting

Python code reference: *Eexp\_rule, Eimp\_rule*

i.e., For each hub and time period:

Non-import energy carrier exports = 0

Non-export energy carrier exports = 0

## Technology energy and capacity constraints

Technology energy balance with shares

Python code reference: *EoutFree\_rule, EoutFix\_rule, EinFix\_rule*

For technologies with NO fixed output shares:

For technologies with FIXED OUTPUT shares:

i.e.,

Energy output = energy input \* efficiency

For technologies with FIXED INPUT shares:

i.e.,

Energy input = total energy input \* energy input share

Energy input/output setting

i.e., set energy input/output to zero for each energy carrier that cannot be input/output from a technology in each hub

Capacity/energy output bound

Python code reference: *capConstFree\_rule, capConstFix*

For technologies with NO fixed output shares:

For technologies with FIXED OUTPUT shares:

i.e., For each hub, time period, technology and energy carrier:

Energy output <= capacity

Maximum capacity bound

Python code reference: *maxCapacity\_rule*

i.e., For each hub, technology and energy carrier:

Capacity <= maximum allowable capacity

Note: maxCapT == 0 for configurations that are not allowed, and maxCapT ==infinity where a configuration is allowed, but max cap has not been specified.

Minimum capacity bound

Python code reference: *minCapacity\_rule*

i.e., For each hub, technology and energy carrier:

Capacity >= minimum allowable capacity

Maximum energy output bound

Python code reference: *maxOutConstFree\_rule, maxOutConstFix*

For technologies with NO fixed output shares:

For technologies with FIXED OUTPUT shares:

i.e., For each technology and hub:

Total energy output <= maximum allowable production

Minimum energy output bound

Python code reference: *minOutConstFree\_rule, minOutConstFix*

For technologies with NO fixed output shares:

For technologies with FIXED OUTPUT shares:

i.e., For each technology and hub:

Total energy output >= minimum allowable production

## Part load constraints

Part load lower bound

Python code reference: *partLoadFreeL\_rule, partLoadFixL*

For technologies with NO fixed output shares:

For technologies with FIXED OUTPUT shares:

Part load upper bound

Python code reference: *partLoadU\_rule*

i.e.,

If a part-load technology is operating (on==1), then the plant must operate at at least the part-load capacity

If part-load technology *plt* is on (1) in time m, then E >= pl\*capacity, and E <= bigM (i.e., infinity);

If the part-load technology is off (0) in time m, then E >= negative bigM (i.e.., infinity), and E <= 0

Note: will be 1 if Eout > 0, but it otherwise has no meaning.

## Solar constraints

Solar irradiation constraint

Python code reference: *solarInput\_rule*

i.e., For each solar technology, hub and time period:

Input solar energy = solar insolation \* capacity / specific power of panel

Maximum rooftop constraint (custom constraint)

Python code reference: *roofArea\_rule*

i.e., For each hub, rooftop technology (i.e., solar technology):

Total rooftop area occupied by installed solar technologies <= total rooftop area

## Network constraints

Network activity constraint

Python code reference: *netQ\_rule*

i.e., For each link at each time period:

Network flow < capacity if connection is possible (i.e., has been specified and flow direction is allowed)

Network operational binary indicator

Python code reference: *YxOp\_rule*

i.e., Binary variable is set to 1 if NetQ > 0. (Note that under other conditions, binary indicator is meaningless).

Ensure one-way flow in each time period

Python code reference: *netflow\_rule*

i.e., Ensure that flow does not go in both directions in a single time period.

Network capacity equality

Python code reference: *netCapEq\_rule*

i.e., Capacity from hub i to j is the same as capacity from hub j to i

Maximum network capacity

Python code reference: *netMaxCap\_rule*

i.e., Capacity is less than maximum capacity for each link.

Note: maxCapN == 0 for configurations that are not allowed, and maxCapN ==infinity where a configuration is allowed, but max cap has not been specified.

Minimum network capacity

Python code reference: *netMinCap\_rule*

i.e., Capacity is greater than minimum capacity for each link.

Network parameter settings

Python code reference: *NetworkInit* method within *input\_data.py*

If connection configuration does not exist for , then:

If link has a uni-directional flow (“y”) from i to j, then:

## Storage constraints

Storage balance

Python code reference: *storageBalance\_rule*

i.e., For each storage technology for each time period, hub and energy carrier:

State of charge (SoC) in time t = standing storage loss rate \* SoC in time t-1 + storage charging efficiency \* energy into storage in time t – energy out of storage in time t / storage discharging efficiency

Storage state-of-charge (SoC) equality

Python code reference: *storageStartEnd\_rule*

i.e., For each hub and storage technology:

Start SoC is equal to the end SoC

Initial state-of-charge

Python code reference: *initSoc\_rule*

i.e., For each hub and storage technology:

Start SoC is equal to minimum SoC required

Storage operational binary indicators

Python code reference: *YstgIn\_rule, YstgOut\_rule*

i.e., Binary variable YNstgIn (YNstgOut) is set to 1 if EinS > 0 (EoutS > 0). (Note that under other conditions, binary indicator is meaningless).

Ensure one-way storage flow in each time period

Python code reference: *stgflow\_rule*

i.e., Ensure that storage does not go in both directions in a single time period.

Maximum charging

Python code reference: *storageChargeRate\_rule*

i.e., For each storage technology in each time period and hub:

Energy into storage <= maximum charging rate \* storage capacity

i.e., energy into storage cannot exceed maximum charging rate

Maximum discharging

Python code reference: *storageDischRate\_rule*

i.e., For each storage technology in each time period and hub:

Energy out of storage <= maximum discharge rate \* storage capacity

i.e., energy into storage cannot exceed maximum charging rate

Storage minimum SoC

Python code reference: *storageMinState\_rule*

i.e., For each hub and storage technology:

Start SoC >= minimum SoC

Storage capacity

Python code reference: *storageCap\_rule*

i.e., For each storage technology in each time period and hub:

State of charge <= storage capacity (kWh)

Maximum storage capacity

Python code reference: *storageMaxCap\_rule*

i.e., For each storage technology in each time period and hub:

Capacity <= maximum allowable capacity.

Note: maxCapS == 0 for configurations that are not allowed, and maxCapS ==infinity where a configuration is allowed, but max cap has not been specified.

Minimum storage capacity

Python code reference: *storageMinCap\_rule*

i.e., For each storage technology in each time period and hub:

Capacity >= maximum allowable capacity.

## Cost constraints

Total cost

Python code reference: *totalCost\_rule*

Investment cost

Python code reference: *invCost\_rule*

Fuel cost

Python code reference: *fuelCost\_rule*

Variable operation and maintenance cost

Python code reference: *vomCost\_rule*

Fixed operation and maintenance cost

Python code reference: *fomCost\_rule*

Carbon tax

Python code reference: *co2Tax\_rule*

Income from export energy

Python code reference: *incomeExp\_rule*

## Carbon constraints

Total carbon

Python code reference: *totalCarbon2\_rule, totalCarbon\_rule*

Maximum carbon emissions

Python code reference: *carbonConst\_rule*

Total carbon emission <= maximum allowable emissions over modeling horizon

CO2 from import energy carrier usage

Python code reference: *fuelCO2\_rule*

CO2 from technology installation

Python code reference: *techCO2\_rule*

CO2 from storage installation

Python code reference: *stgCO2\_rule*

CO2 from network link installation

Python code reference: *netCO2\_rule*