Important: This document resumes the integrations performed to the available Temoa source code (TemoaProject/temoa: Tools for Energy Model Optimization and Analysis (github.com)) to obtain the integrated MAHTEP version (MAHTEP/TEMOA-Italy (github.com)). Two kinds of integrations have been performed: creating new tables within the .sql database; adding pieces of code to the Temoa source code. The document indicates the exact .py file and the exact row where the pieces of code have been modified.

EMISSION FACTORS

Create a new table in the **database** with the following structure:

```
CREATE TABLE "CommodityEmissionFactor" (
      "input_comm"    text,
      "emis_comm"     text,
      "ef"            real,
      "emis_unit"     text,
      "ef_notes"      text,
      PRIMARY KEY("input_comm","ef","emis_comm"),
      FOREIGN KEY("input_comm") REFERENCES "commodities"("comm_name"),
      FOREIGN KEY("emis_comm") REFERENCES "commodities"("comm_name")
);
```

SERVICE DEMANDS PROJECTION

Create new tables in the **database** with the following structure:

```
CREATE TABLE "Driver" (
    "regions"       text,
    "periods"     integer,
      "driver_name"     text,
      "driver"        real,
      "driver_notes"  text,
      PRIMARY KEY("regions", "periods", "driver_name"),
      FOREIGN KEY("regions") REFERENCES "regions"("regions"),
      FOREIGN KEY("periods") REFERENCES "time_periods"("t_periods")
);

CREATE TABLE "Allocation" (
    "regions"       text,
      "demand_comm"     text,
      "driver_name"     text,
      "allocation_notes"  text,
      PRIMARY KEY("regions", "demand_comm", "driver_name"),
      FOREIGN KEY("regions") REFERENCES "regions"("regions"),
      FOREIGN KEY("demand_comm") REFERENCES "commodities"("comm_name"),
      FOREIGN KEY("driver_name") REFERENCES "Driver"("driver_name"),
);

CREATE TABLE "Elasticity" (
    "regions"       text,
    "periods"     integer,
      "demand_comm"     text,
      "elasticity"    real,
      "elaticity_notes"  text,
      PRIMARY KEY("regions", "periods", "demand_comm"),
      FOREIGN KEY("regions") REFERENCES "regions"("regions"),
      FOREIGN KEY("periods") REFERENCES "time_periods"("t_periods"),
      FOREIGN KEY("demand_comm") REFERENCES "commodities"("comm_name")
);
```

CAPACITY FACTOR

Create a new table in the **database** with the following structure:

```
CREATE TABLE "CapacityFactor" (
	"regions"	text,
	"tech"	text,
	"vintage"	integer,
	"cf"	real,
	"cf_notes"	text,
	PRIMARY KEY("regions","tech","vintage"),
	FOREIGN KEY("tech") REFERENCES "technologies"("tech"),
	FOREIGN KEY("vintage") REFERENCES "time_periods"("t_periods")
);
```

Add to **temoa_config.py** at row 157 the following rows:

```
            ['param','CapacityFactor',            '',                     '',
3],
```

Add to **temoa_initialize.py** at row 798 the following rows:

```
def CapacityFactorIndices(M):
	indices = set(
		(r, t, v)

		for r in M.regions
		for t in M.tech_all
		for v in M.vintage_all
	)

	return indices
```

Add to **temoa_model.py** at row 141 the following rows:

```
    M.CapacityFactor_rtv = Set(dimen=3, initialize= CapacityFactorIndices)
    M.CapacityFactor = Param(M.CapacityFactor_rtv, default=1)
```

Add to **temoa_rules.py** at row 82 the following row:

```
    * value(M.CapacityFactor[r, t, v]) \
```

Add to **temoa_rules.py** at row 91 the following row:

```
    * value(M.CapacityFactor[r, t, v]) \
```

Add to **temoa_rules.py** at row 133 the following row:

```
    * value(M.CapacityFactor[r, t, v]) \
```

MAX ACTIVITY GROUP

Create new tables in the **database** with the following structure:

```
CREATE TABLE "MaxGenGroupWeight" (
      "regions"   text,
      "tech"      text,
      "max_group_name"  text,
      "act_fraction"    REAL,
      "notes"     text,
      PRIMARY KEY("tech","max_group_name","regions")
);
CREATE TABLE "MaxGenGroupLimit" (
      "periods"   integer,
      "max_group_name"  text,
      "max_act_g" real,
      "notes"     text,
      PRIMARY KEY("periods","max_group_name")
);
```

Add to **temoa_config.py** at row 133 the following rows:

```
            ['param','MaxGenGroupLimit',          '',                     '',
2],
            ['param','MaxGenGroupWeight',         '',                     '',
3],
```

Add to **temoa_model.py** at row 225 the following rows:

```
    M.MaxGenGroupWeight = Param(M.RegionalIndices, M.tech_groups, M.groups,
default=0)
    M.MaxGenGroupLimit = Param(M.time_optimize, M.groups)
```

Add to **temoa_model.py** at row 454 the following rows:

```

    M.MaxActivityGroup_pg = Set(
        dimen=2, initialize=lambda M: M.MaxGenGroupLimit.sparse_iterkeys()
    )
    M.MaxActivityGroup = Constraint(
        M.MaxActivityGroup_pg, rule=MaxActivityGroup_Constraint
    )
```

Add to **temoa_rules.py** at row 1744 the following rows:

```

def MaxActivityGroup_Constraint(M, p, g):
    r"""

The MaxActivityGroup constraint sets a maximum activity limit for a user-
defined
technology group. Each technology within each group is multiplied by a
weighting function, which determines what technology activity share can count
towards the constraint.

.. math::
```

```
   :label: MaxActivityGroup

       \sum_{S,D,I,T,V,O} \textbf{FO}_{p, s, d, i, t, v, o} \cdot WEIGHT_{t|t
\not \in T^{a}}
       + \sum_{I,T,V,O} \textbf{FOA}_{p, i, t, v, o} \cdot WEIGHT_{t \in
T^{a}}
       \le MGGL_{p, g}

       \forall \{p, g\} \in \Theta_{\text{MaxActivityGroup}}

where :math:`g` represents the assigned technology group and :math:`MGGL`
refers to the :code:`MaxGenGroupLimit` parameter.
"""


    activity_p = sum(
        M.V_FlowOut[r, p, s, d, S_i, S_t, S_v, S_o] * M.MaxGenGroupWeight[r,
S_t, g]
        for r in M.RegionalIndices
        for S_t in M.tech_groups if (S_t not in M.tech_annual) and ((r, p,
S_t) in M.processVintages.keys())
        for S_v in M.processVintages[r, p, S_t]
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    activity_p_annual = sum(
        M.V_FlowOutAnnual[r, p, S_i, S_t, S_v, S_o] * M.MaxGenGroupWeight[r,
S_t, g]
        for r in M.RegionalIndices
        for S_t in M.tech_groups if (S_t in M.tech_annual) and ((r, p, S_t)
in M.processVintages.keys())
        for S_v in M.processVintages[r, p, S_t]
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
    )

    max_act = value(M.MaxGenGroupLimit[p, g])
    expr = activity_p + activity_p_annual <= max_act
    return expr
```

MIN/MAX INPUT/OUTPUT GROUP

Create new tables in the **database** with the following structure:

```
CREATE TABLE "MinInputGroupWeight" (
      "regions"           text,
      "tech"                    text,
      "group_name"          text,
      "gi_min_fraction" real,
      "notes"             text,
      PRIMARY KEY("tech","group_name","regions")
);
CREATE TABLE "MinInputGroup" (
      "regions"         text,
      "periods"         integer,
      "input_comm"        text,
      "group_name"        text,
      "gi_min"          real,
      "gi_min_notes"    text,
      FOREIGN KEY("group_name") REFERENCES "groups"("group_name"),
      FOREIGN KEY("input_comm") REFERENCES "commodities"("comm_name"),
      FOREIGN KEY("periods") REFERENCES "time_periods"("t_periods"),
      PRIMARY KEY("regions","periods","input_comm","group_name")
);
CREATE TABLE "MaxInputGroupWeight" (
      "regions"           text,
      "tech"                    text,
      "group_name"          text,
      "gi_max_fraction" real,
      "notes"             text,
      PRIMARY KEY("tech","group_name","regions")
);
CREATE TABLE "MaxInputGroup" (
      "regions"         text,
      "periods"         integer,
      "input_comm"        text,
      "group_name"        text,
      "gi_max"          real,
      "gi_max_notes"    text,
      FOREIGN KEY("group_name") REFERENCES "groups"("group_name"),
      FOREIGN KEY("input_comm") REFERENCES "commodities"("comm_name"),
      FOREIGN KEY("periods") REFERENCES "time_periods"("t_periods"),
      PRIMARY KEY("regions","periods","input_comm","group_name")
);
CREATE TABLE "MaxOutputGroupWeight" (
      "regions"           text,
      "tech"                    text,
      "group_name"          text,
      "go_max_fraction" real,
      "notes"             text,
      PRIMARY KEY("tech","group_name","regions")
);
CREATE TABLE "MaxOutputGroup" (
      "regions"         text,
```

```
      "periods"         integer,
      "output_comm"       text,
      "group_name"        text,
      "go_max"          real,
      "go_max_notes"    text,
      FOREIGN KEY("group_name") REFERENCES "groups"("group_name"),
      FOREIGN KEY("output_comm") REFERENCES "commodities"("comm_name"),
      FOREIGN KEY("periods") REFERENCES "time_periods"("t_periods"),
      PRIMARY KEY("regions","periods","output_comm","group_name")
);
```

Add to **temoa_config.py** at row 149 the following rows:

```
            ['param','MinInputGroupWeight',       '',                   '',
3],
            ['param','MinInputGroup',             '',                   '',
4],
            ['param','MaxInputGroupWeight',       '',                   '',
3],
            ['param','MaxInputGroup',             '',                   '',
4],
            ['param','MaxOutputGroupWeight',      '',                   '',
3],
            ['param','MaxOutputGroup',            '',                   '',
4],
```

Add to **temoa_model.py** at row 166 the following rows:

```
    M.MinInputGroupWeight = Param(M.regions, M.tech_groups, M.groups,
default=0)
    M.MinInputGroup = Param(M.regions, M.time_optimize, M.commodity_physical,
M.groups)
    M.MaxInputGroupWeight = Param(M.regions, M.tech_groups, M.groups,
default=0)
    M.MaxInputGroup = Param(M.regions, M.time_optimize, M.commodity_physical,
M.groups)
    M.MaxOutputGroupWeight = Param(M.regions, M.tech_groups, M.groups,
default=0)
    M.MaxOutputGroup = Param(M.regions, M.time_optimize,
M.commodity_physical, M.groups)
```

Add to **temoa_model.py** at row 545 the following rows:

```
    M.MinInputGroup_Constraint_rpig = Set(
        dimen=4, initialize=lambda M: M.MinInputGroup.sparse_iterkeys()
    )
    M.MinInputGroupConstraint = Constraint(
        M.MinInputGroup_Constraint_rpig, rule=MinInputGroup_Constraint
    )

    M.MaxInputGroup_Constraint_rpig = Set(
        dimen=4, initialize=lambda M: M.MaxInputGroup.sparse_iterkeys()
    )
    M.MaxInputGroupConstraint = Constraint(
```

```
        M.MaxInputGroup_Constraint_rpig, rule=MaxInputGroup_Constraint
    )

    M.MaxOutputGroup_Constraint_rpig = Set(
        dimen=4, initialize=lambda M: M.MaxOutputGroup.sparse_iterkeys()
    )
    M.MaxOutputGroupConstraint = Constraint(
        M.MaxOutputGroup_Constraint_rpig, rule=MaxOutputGroup_Constraint
    )
```

Add to **temoa_rules.py** at row 2097 the following rows:

```
def MinInputGroup_Constraint(M, r, p, i, g):
    r"""

Allows users to specify minimum shares of commodity inputs to a group of
technologies.
These shares can vary by model time period.

"""
    inp = sum(
        M.V_FlowOut[r, p, s, d, i, S_t, S_v, S_o] * M.MinInputGroupWeight[r,
S_t, g] / value(M.Efficiency[r, i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v] if S_i == i
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    inp_annual = sum(
        M.V_FlowOutAnnual[r, p, i, S_t, S_v, S_o] * M.MinInputGroupWeight[r,
S_t, g] / value(M.Efficiency[r, i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v] if S_i == i
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
    )

    total_inp = sum(
        M.V_FlowOut[r, p, s, d, S_i, S_t, S_v, S_o] *
M.MinInputGroupWeight[r, S_t, g] / value(M.Efficiency[r, S_i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
```

```
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    total_inp_annual = sum(
        M.V_FlowOutAnnual[r, p, S_i, S_t, S_v, S_o] *
M.MinInputGroupWeight[r, S_t, g] / value(M.Efficiency[r, S_i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
    )

    min_inp = value(M.MinInputGroup[r, p, i, g])
    expr = (inp + inp_annual) >= min_inp * (total_inp + total_inp_annual)
    return expr

def MaxInputGroup_Constraint(M, r, p, i, g):
    r"""

Allows users to specify maximum shares of commodity inputs to a group of
technologies.
These shares can vary by model time period.

"""
    inp = sum(
        M.V_FlowOut[r, p, s, d, i, S_t, S_v, S_o] * M.MaxInputGroupWeight[r,
S_t, g] / value(M.Efficiency[r, i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v] if S_i == i
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    inp_annual = sum(
        M.V_FlowOutAnnual[r, p, i, S_t, S_v, S_o] * M.MaxInputGroupWeight[r,
S_t, g] / value(M.Efficiency[r, i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v] if S_i == i
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
```

```
    )

    total_inp = sum(
        M.V_FlowOut[r, p, s, d, S_i, S_t, S_v, S_o] *
M.MaxInputGroupWeight[r, S_t, g] / value(M.Efficiency[r, S_i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    total_inp_annual = sum(
        M.V_FlowOutAnnual[r, p, S_i, S_t, S_v, S_o] *
M.MaxInputGroupWeight[r, S_t, g] / value(M.Efficiency[r, S_i, S_t, S_v, S_o])
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
    )

    max_inp = value(M.MaxInputGroup[r, p, i, g])
    expr = (inp + inp_annual) <= max_inp * (total_inp + total_inp_annual)
    return expr

def MaxOutputGroup_Constraint(M, r, p, o, g):
    r"""

Allows users to specify maximum shares of commodity outputs to a group of
technologies.
These shares can vary by model time period.

"""
    outp = sum(
        M.V_FlowOut[r, p, s, d, S_i, S_t, S_v, o] * M.MaxOutputGroupWeight[r,
S_t, g]
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i] if S_o == o
        for s in M.time_season
        for d in M.time_of_day
    )

    outp_annual = sum(
```

```
        M.V_FlowOutAnnual[r, p, S_i, S_t, S_v, o] * M.MaxOutputGroupWeight[r,
S_t, g]
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i] if S_o == o
    )

    total_outp = sum(
        M.V_FlowOut[r, p, s, d, S_i, S_t, S_v, S_o] *
M.MaxOutputGroupWeight[r, S_t, g]
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t not in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
        for s in M.time_season
        for d in M.time_of_day
    )

    total_outp_annual = sum(
        M.V_FlowOutAnnual[r, p, S_i, S_t, S_v, S_o] *
M.MaxOutputGroupWeight[r, S_t, g]
        for S_r, S_p, S_t, S_v in M.activeActivity_rptv
        if S_r == r
        if S_p == p
        if S_t in M.tech_groups
        if S_t in M.tech_annual
        for S_i in M.processInputs[r, p, S_t, S_v]
        for S_o in M.ProcessOutputsByInput[r, p, S_t, S_v, S_i]
    )

    max_outp = value(M.MaxOutputGroup[r, p, o, g])
    expr = (outp + outp_annual) <= max_outp * (total_outp +
total_outp_annual)
    return expr
```

MAX RESOURCE

This constraint has been modified with respect to the previous version. In particular, the MaxResource_Constraint definition has been modified in **temoa_rules.py**, adding the following expression at the end of rows 1874 and 1885.

This is to consider in the computation the different lengths of the time periods.

```
  * M.PeriodLength[p]
```