MGET GAMS Model: Explanation of the Load Processing Script

Contents

1 Nomenclature

1.1 Sets

Symbol	Description
Y	Set of years in the planning horizon
H	Set of operational hours
E	Set of energy carriers (e.g., Gas, Hydrogen, Electricity)
F	Set of fuels (e.g., Natural Gas, Hydrogen)
N	Set of nodes (geographical locations, NUTS3 level)
$NUTS2$	Set of NUTS2 regions (aggregated nodes)
RGN	Set of larger regions
CN	Set of countries
A	Set of arcs (pipeline or transmission connections between nodes)
Z	Set of surplus/deficit penalty types
aux_tot	Auxiliary set for label ordering (e.g., TOT, Obj, Idx)
aux_rep	Auxiliary set for reporting labels (e.g., bidirectional flows, repurposing)
aux_rep_c	Auxiliary set for reporting components (e.g., storages, blending)
$A_s(a,n)$	Set mapping arcs to their start node
$A_e(a,n)$	Set mapping arcs to their end node
$anm(a,n,m)$	Set indicating an arc exists between node n and node m
$opp(a,a)$	Set of opposite arcs for reversible connections
$n\_in\_c(n,c)$	Set mapping nodes to countries
$n\_in\_2(n,NUTS2)$	Set mapping nodes to NUTS2 regions
$n\_in\_r(n,RGN)$	Set mapping nodes to larger regions
$ypred(y2,y)$	Set defining immediate predecessor years
$yscai(y2,y)$	Set defining successor years (including current year)

1.2 Indices

Symbol	Description
y	Index for years ($y \in Y$)
$y2$	Index for predecessor or successor years ($y2 \in Y$)
h	Index for operational hours ($h \in H$)
$h2$	Index for operational hours (alias for h)
e	Index for energy carriers ($e \in E$)
f	Index for fuels ($f \in F$) or secondary energy carriers

n	Index for nodes ($n \in N$)
m	Index for nodes (used for destination nodes)
a	Index for arcs ($a \in A$)
ao	Index for arcs (opposite direction)
ai	Index for arcs (original direction)
c	Index for countries ($c \in CN$)
cn	Alias for countries
rgn	Index for regions ($rgn \in RGN$)
ro	Alias for regions (origin)
ri	Alias for regions (destination)
z	Index for penalty types ($z \in Z$)
$yrep$	Index for years in repetition ($yrep \in Y$)
$yrep2$	Alias for $yrep$

1.3 Parameters

Symbol	**Description**
$c_p(n, e, y)$	Cost per unit of gas production at node n for energy carrier e in year y [€/GWh]
$c_a(a, e, y)$	Cost of transporting one unit of energy carrier e through arc a in year y [€/GWh/100km]
$c_{ab}(a, e, y)$	Cost per unit of bidirectional arc capacity for energy carrier e on arc a in year y [€/GWh/100km]
$c_{ax}(a, e, y)$	Cost per unit of expanding capacity for arc a, energy carrier e in year y [€/GWh/100km/year]
$c_z(z, e)$	Penalty cost per unit of unmet demand for energy carrier e in shortage type z [€/GWh]
$c_{we}(n, e)$	Cost per unit of gas extracted from storage at node n for energy carrier e [€/GWh]
$c_{lr}(n, e)$	Cost per unit of regasification at node n for energy carrier e [€/GWh]
$c_{bl}(e, f)$	Cost per unit of blending energy carrier e into fuel f [€/GWh]
$cap_{wi}(n, e, y)$	Maximum gas injection capacity at node n for energy carrier e in year y [GWh]
$cap_{we}(n, e, y)$	Maximum gas extraction capacity at node n for energy carrier e in year y [GWh]
$cap_p(n, e, y, h)$	Maximum production capacity at node n for energy carrier e in year y and hour h [GWh]
$cap_{ww}(n, e, y)$	Maximum working storage capacity at node n for energy carrier e in year y [GWh]
$cap_a(a, e, y)$	Arc capacity flow limit for energy carrier e through arc a in year y [GWh]
$dmd(n, e, y, h)$	NUTS3 level gas demand at node n for energy carrier e in year y and hour h [GWh]
$dmd2(nuts2, e, y, h)$	NUTS2 level gas demand for energy carrier e in region $nuts2$ in year y and hour h [GWh]
$e_a(a, e)$	Efficiency of transporting energy carrier e along arc a [%]

$e_w(n,e)$	Storage efficiency at node n for energy carrier e [%]
$EOH(y)$	End of horizon adjustment value for year y [dimensionless]
$f_{ar}(a,e,f,y)$	Fixed cost for repurposing arc a from energy carrier e to f in year y [€/GWh]
$f_{ab}(a,y)$	Fixed cost of converting arc a to bidirectional operation in year y [€]
$lb_p(n,e,y,h)$	Minimum production (lower bound) at node n for energy carrier e in year y, hour h [GWh]
$ub_{bl}(e,f)$	Upper bound blending fraction for energy carrier e into fuel f [%]
$lb_{ax}(a,e,y)$	Lower bound on planned expansion for arc a and energy carrier e in year y [GWh]
$ub_{ax}(a,y)$	Upper bound on expansion for arc a in year y [GWh]
$stor_i(n,e,y,h,*)$	Storage injection profile at node n for energy carrier e, year y, hour h [GWh]
$stor_x(n,e,y,h,*)$	Storage extraction profile at node n for energy carrier e, year y, hour h [GWh]
$bigM$	Big constant for logical constraints (e.g., binary-enforcing parameters) [GWh]
$is\_bid(a)$	Indicator if arc a is bidirectional [1]
$r(y)$	Discount factor for year y [1]
$scaleUp(h)$	Scaling factor for the number of hours represented by operational hour h [1]
$vola2(e)$	Volumetric adjustment factor for arcs for energy carrier e [1]
$vols2(e)$	Volumetric adjustment factor for storage volume for energy carrier e [1]
$ypred(y2,y)$	Immediate predecessor year relationship [binary, 0 or 1]
$yscai(y2,y)$	All successor years relationship (including current year) [binary, 0 or 1]

1.4 Decision Variables

1.4.1 Binary Decision Variables

Symbol	Description
$B_{BD}(a,y)$	Binary indicator for decision to make arc a bidirectional in year y [1]
$B_{AR}(a,e,f,y)$	Binary indicator for repurposing arc a from energy carrier e to f in year y [1]
$B_{WR}(n,e,f,y)$	Binary indicator for repurposing storage at node n from carrier e to f in year y [1]

1.4.2 Continuous Decision Variables

Symbol	Description
TC	Total system cost (objective function) [€]
$F_A(a,e,y,h)$	Flow of energy carrier e through arc a in year y, hour h [GWh]
$K_A(a,e,y)$	Arc capacity for energy carrier e through arc a in year y [GWh]
$K_W(n,e,y)$	Working storage capacity at node n for energy carrier e in year y [GWh]
$K_{BD}(a,e,y)$	Capacity allocated to bidirectional flow on arc a for energy carrier e in year y [GWh]

$K_{OPP}(a, e, y)$	Capacity for reversed (opposite) direction flow on arc a for energy carrier e in year y [GWh]
$K_{RA}(a, e, f, y)$	Repurposed capacity from energy carrier e to f on arc a in year y [GWh]
$K_{RW}(n, e, f, y)$	Repurposed storage capacity at node n from carrier e to f in year y [GWh]
$Q_P(n, e, y, h)$	Quantity of energy carrier e produced at node n in year y and hour h [GWh]
$Q_E(n, e, y, h)$	Quantity extracted from storage at node n for energy carrier e in year y, hour h [GWh]
$Q_I(n, e, y, h)$	Quantity injected into storage at node n for energy carrier e in year y, hour h [GWh]
$Q_R(n, e, y, h)$	Amount of LNG regasified at node n for energy carrier e in year y, hour h [GWh]
$Q_B(n, e, f, y, h)$	Amount of energy carrier e blended into fuel f at node n in year y, hour h [GWh]
$Q_S(n, e, y, h)$	Quantity of demand met (sales) for energy carrier e at node n in year y, hour h [GWh]
$X_A(a, e, y)$	Arc capacity expansion for energy carrier e along arc a in year y [GWh]
$ZDS(z, n, e, y, h)$	Unmet gas demand at node n in shortage zone z for energy carrier e in year y, hour h [GWh]
$ZN2(nuts2, e, y, h)$	Unmet gas demand in region $nuts2$ for energy carrier e in year y, hour h [GWh]

2 Introduction

The `MGET_v0_2_0.gms` model requires a structured data-loading process to incorporate various input parameters and scenario specifications. The main script responsible for handling this process is `load_input_from_Excel.gms`, which is versioned internally and compatible with model version `v0.2.0`. This script facilitates the loading of input data from an external Excel file, `Spain.xlsx`, located in the `cases/Spain_case/input_data/` folder.

Additionally, the model uses multiple scenario-based input files in the same directory to define specific case studies and operational assumptions:

- `2040.gms` → Defines the base-year time horizon.
- `scen_Spain_2040_4_0_dmd.gms` → Moderate demand scenario.
- `scen_Spain_2040_4_0_sup.gms` → Moderate supply scenario.
- `scen_Spain_2040_4_2_dmd.gms` → Alternative demand scenario.
- `scen_Spain_2040_4_2_sup.gms` → Alternative supply scenario.

To ensure correctness of the input network, the model includes a verification step using `Verify_nodes_arcs.gms`, which checks consistency in node-arc definitions. This script identi-

fies and flags errors such as unconnected nodes, missing arcs, or infeasible configurations, and prevents the model from proceeding if critical issues are found.

Purpose of the Data-Loading Process: The data-loading process enables seamless integration of external Excel datasets into the GAMS environment. It automates the conversion to GDX format, dynamically includes scenario files, and ensures all required sets and parameters are initialized before model execution. This modular approach reduces manual input, supports flexible scenario analysis, and ensures coherence between input files and model logic. It also facilitates iteration and extension of the model without changes to the core code.

3 Reading Input Data from Excel

The script `load_input_from_Excel.gms` (version `v0.2.0`) automates the conversion of input data from an Excel workbook into GAMS-readable GDX files. This is achieved using the GAMS utility `GDXXRW`, which reads named ranges defined in the Excel file.

GAMS Code Snippet

```
$call "%KeepGdx%GDXXRW %excel_file% O=gdx/%string%_inputs SkipEmpty=0 @read_data.txt"
$call "%KeepGdx%GDXXRW %excel_file% O=gdx/%data% SkipEmpty=0 @read_data.txt"
$ifE errorLevel<>0 $abort.noError "Something went wrong reading Excel. Double check named
ranges or Excel save."
```

These commands:

- Convert the Excel file `%excel_file%` (e.g., `Spain.xlsx`) into two GDX files:
  - `gdx/%string%_inputs.gdx` — full structured input for the model run
  - `gdx/%data%.gdx` — raw version for debugging and validation
- Back up any previous GDX file versions before overwriting
- Use the mapping file `read_data.txt` to define which named ranges to extract

The `%KeepGdx%` macro optionally suppresses GDX file regeneration — this is especially useful during debugging or when the GDX files are already up-to-date. If `GDXXRW` encounters an error (e.g., due to missing or unsaved named ranges in the Excel workbook), the script halts execution using the `$abort.noError` directive and displays a descriptive error message.

The resulting GDX files are then used to load sets and parameters such as A, N, F, and structured parameters like dat_o, dat_p, and dat_a, all of which are essential for building the GAMS model input structure.

Version Reference: This process is implemented in load_input_from_Excel.gms, version v0.2.0, and is compatible with model version MGET_v0_2_0.gms.

3.1 Defining Sets and Aliases

The script defines core sets to structure the model's data and topology, covering nodes, arcs, regions, fuels, and deviation categories. These sets are fundamental to organizing the input structure and enforcing network constraints in the gas system model. Aliases are used to simplify looping and summations in the model logic — especially when working with directional arcs, fuels, or regional aggregates.

Core Sets and Aliases

```
Set
    aux_tot     "Auxiliary set for label order [-]"    /TOT,Obj,Idx,''/
    A           "Arcs; if reversible define the other direction too [-]"
    CN          "Countries [-]"
    F           "Fuels / Energy carriers [-]"
    N           "Nodes (NUTS3 level) [-]"
    NUTS2       "NUTS2 regions [-]"
    RGN         "Regions [-]"
    Z           "Surplus / deficit penalty types [-]"
;
alias (a,ao,ai),(e,f),(n,m),(rgn,ro,ri),(cn,c);
```

Auxiliary Classification Sets

Additional sets aux_rep and aux_rep_c are defined to assist with structured labeling, consistent ordering, and group-specific behaviors across model parameters. These are especially important for automated reporting, repurposing options, expansion logic, and constraint generation.

```
Set
    aux_rep "Auxiliary set for label order"
        /TOT,nom,disc,'+','-','Z','K','0',
        'purp-','purp+',purp,bidir,'cap',capBD,1,2,3,4,5,6,7,8,'expans',
        P,D,D2,LNG,A,A+,A-,'C','G','H',W,P+,M+,Z+,P-,Z-,D-/
```

```
    aux_rep_c
        /prod,flow+,flow-,expans+,expans-, purp,repurp,bidir,bidfx,
        invest,Stor,Regas,Blend,ZXA,ZN2,ZA,ZD,ZMD,ZMS,ZPL,ZPU/
```

These sets enable modular referencing across different model modules and ensure consistency when referencing parameter categories (e.g., for blending, repurposing, or hydrogen/gas technologies).

3.2 Definition of Time-Related Sets

The model defines the time structure and associated parameters to represent operational hours, multi-year planning horizons, and long-term discounting. These elements are critical to accurately modeling investment decisions and intertemporal trade-offs in infrastructure and supply-demand flows.

Time Sets and Parameters

```
$include "%path%/input_data/%hor%.gms"              /* Load yearly planning horizon */
set H hours /1*%oper%/;

ALIAS (y,y2),(yrep,yrep2),(h,h2);

Parameter
    ypred(y,y)      "Immediate predecessor year"
    yscai(y,y)      "All successors including the current year"
    EOH(y)          "End of Horizon correction"
;

ypred(y2,y)$(ord(y)-ord(y2)=1) = 1;
yscai(y2,y)$(ord(y) <= ord(y2)) = 1;

EOH(y) = 1;
EOH(y)$(ord(y) = card(y)) = 3;
```

The script dynamically loads the set of model years using a parameterized include path: `%hor%` is typically defined in the configuration file to refer to a GAMS script such as `2040.gms`, containing the set of planning years.

The set `H` defines the hourly resolution for operations (e.g., 1 to 8760 for full-year hourly resolution). The aliases `y2`, `h2`, etc., enable recursive or pairwise logic used in intertemporal constraints.

Discounting and Horizon Correction: Two key time-related parameters are introduced:

- `ypred(y,y)`: defines the immediate predecessor of each year.
- `yscai(y,y)`: includes all years equal to or after the current year.

The parameter `EOH(y)` introduces an adjustment for the final year of the model horizon (set to 3), reflecting either an extrapolation assumption or a terminal value correction in the investment model.

3.3 Loading and Assigning Global Parameters

The model loads general configuration data from the Excel sheet named `other_data` into the parameter `dat_o`. This sheet contains scalar constants, penalty terms, and metadata used across the model. The following parameters are computed or extracted from `dat_o`.

GAMS Code Snippet

```
Parameter
    dat_o        "Raw data loaded from Excel (sheet: 'other_data')"
    bigM         "Big-M constant for logical constraints [GWh/h]"
    c_bl         "Blending cost: fuel e into f [€/GWh]"
    c_z(z,e)     "Feasibility penalties for surplus/deficit type z, fuel e [€/GWh]"
    is_g(e)      "Flag: 1 if natural gas, 0 otherwise [-]"
    is_h(e)      "Flag: 1 if hydrogen, 0 otherwise [-]"
    not_g(e)     "Flag: 1 if not natural gas [-]"
    not_h(e)     "Flag: 1 if not hydrogen [-]"
    r(y)         "Discount factor by year [-]"
    vola2(e)     "Arc volume unit conversion factor [volume/GWh]"
    vols2(e)     "Storage volume unit conversion factor [volume/GWh]"
    scaleUp(h)   "Number of hours represented by hour slice h [-]"
;

$gdxin gdx/%data%
$load dat_o, f, z

dat_o('BlendCost',e,f)$(dat_o('BlendCost',e,f) <= 0) = dat_o('BlendCost','','');
c_bl(e,f) = dat_o('BlendCost',e,f);

is_g('G') = 1;
is_h('H') = 1;
```

```
not_g(e) = 1; not_g('G') = 0;
not_h(e) = 1; not_h('H') = 0;

bigM = dat_o('bigM','','');

dat_o('Penalty',z,e)$(dat_o('Penalty',z,e) <= 0) = dat_o('Penalty',z,'');
dat_o('Penalty',z,e)$(dat_o('Penalty',z,e) <= 0) = dat_o('Penalty','','');
c_z(z,e) = dat_o('Penalty',z,e);

r(y) = 1 / power(1 + dat_o('DiscRate','',''), dat_o('YearStep','','') * (ORD(y) - 1));
vola2(e) = 1; vola2(e)$dat_o('vola2',e,'') = dat_o('vola2',e,'');
vols2(e) = 1; vols2(e)$dat_o('vols2',e,'') = dat_o('vols2',e,'');
scaleUp(h) = dat_o('scale',h,'');
```

Discount Factor: The discount factor $r(y)$ is calculated as:

$$r(y) = \frac{1}{(1 + \text{DiscRate})^{\text{YearStep} \cdot (\text{ord}(y)-1)}} \tag{1}$$

where `DiscRate` and `YearStep` are scalar values from the Excel sheet. This ensures that future costs are appropriately discounted.

Fuel Classification Flags: The fuel codes are classified using binary flags:

- `is_g('G') = 1` — natural gas is marked as gas
- `is_h('H') = 1` — hydrogen is marked as hydrogen
- `not_g(e) = 1` for all $e \neq$ 'G'
- `not_h(e) = 1` for all $e \neq$ 'H'

Volume Conversion Factors: `vola2(e)` and `vols2(e)` define energy-to-volume conversions for pipeline and storage use. They are initialized to 1 by default and overwritten where defined in the Excel data.

Hour Scaling: The `scaleUp(h)` parameter represents how many real-world hours are covered by each representative time slice, used to weight operating periods correctly in the model.

3.4 Loading and Assigning Node-Level Data

This section of the model processes node-specific input data related to supply, demand, and geographic mappings. It loads structured parameters from the Excel file via GDX and applies logic for regional classification, unit cost assignment, and capacity dynamics.

GAMS Code Snippet

```
Parameter
    cap_p(n,e,y,h)      "Supply capacity at node n [GWh/h]"
    c_lr                "Regasification costs at node n [€/GWh]"
    c_p(n,e,y)          "Supply unit cost [€/GWh]"
    dat_c               "Demand data loaded from Excel"
    dat_n               "Nodes data loaded from Excel"
    dat_p               "Supply data loaded from Excel"
    dat_r               "Regasifier data loaded from Excel"
    dmd(n,e,y,h)        "NUTS3 demand (potential) [GWh/h]"
    dmd2(nuts2,e,y,h)   "NUTS2-level hydrogen demand [GWh/h]"
    lb_p(n,e,y,h)       "Lower bound on supply at node n [GWh/h]"
    ub_bl(e,f)          "Upper blending limit: fuel e into f [%]"
    n_in_c(n,c)         "Mapping: node n in country c [-]"
    n_in_2(n,nuts2)     "Mapping: node n in NUTS2 region [-]"
    n_in_r(n,rgn)       "Mapping: node n in macro-region [-]"
;

$gdxin gdx/%data%
$load n, cn, rgn, nuts2, dat_n, dat_p, dat_r, dat_c
```

Node-to-Region Mapping

Each node is mapped to a country, NUTS2 region, and macro-region using coordinate-based logic:

$$n\_in\_2(n, \text{nuts2}) = \begin{cases} 1, & \text{if } \sum_{c,rgn} |\texttt{LAT}| > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Similar logic is used to define $n\_in\_c(n, c)$ and $n\_in\_r(n, \text{rgn})$, enabling aggregation and filtering at multiple spatial levels.

Supply Unit Cost Assignment

Supply cost `c_p(n,e,y)` is initially assigned using marginal cost (`MC`) values from Excel:

$$c_p(n,e,y) = \begin{cases} \texttt{dat\_p}(n,e,y,\texttt{MC}), & \text{if } y \leq 2030 \\ \texttt{dat\_p}(n,e,2030,\texttt{MC}), & \text{otherwise} \end{cases} \tag{3}$$

This structure ensures continuity of cost assumptions beyond 2030.

Supply Capacity Assignment

Capacities `cap_p(n,e,y,h)` are read from the first hour of the Excel time series:

$$cap_p(n,e,y,h) = \texttt{dat\_p}(n,e,y,'1') \tag{4}$$

Additionally, for the fuel code `g` (natural gas), a fixed value from 2025 is propagated forward:

$$cap_p(n,'g',y,h) = \texttt{dat\_p}(n,'g',2025,'1') \quad \text{for } y \geq 2025 \tag{5}$$

Algerian Gas Reduction For the Algerian node (`DZ000`), a declining supply factor is applied to reflect supply uncertainty:

$$cap_p(\texttt{DZ000},'G',y,h) = \begin{cases} 1.0 \cdot cap_p('DZ000','G',2025,h), & y = 2025 \\ 0.8 \cdot cap_p('DZ000','G',2025,h), & y = 2030 \\ 0.7 \cdot cap_p('DZ000','G',2025,h), & y = 2035 \\ 0.6 \cdot cap_p('DZ000','G',2025,h), & y = 2040 \\ 0.5 \cdot cap_p('DZ000','G',2025,h), & y \geq 2045 \end{cases} \tag{6}$$

Lower Bound for Supply

The minimum allowed supply per node and time step is taken from the Excel column `LB`:

$$lb_p(n,e,y,h) = \texttt{dat\_p}(n,e,y,\texttt{LB}) \tag{7}$$

This constraint ensures that supply decisions adhere to contractual or technical minimums.

3.5 Hydrogen Supply Potential

The model incorporates exogenous projections of hydrogen production potential across multiple nodes and years. These projections reflect anticipated capacity growth in renewable generation and export hubs. Table 6 summarizes the hydrogen supply potential (in normalized units) and associated marginal costs (MC) at each location.

Table 6: Projected Hydrogen Supply Potential by Node and Year

Node	**2035**	**2040**	**2045**	**2050**	**MC [€/GWh]**
PT16E	0.1	0.2	0.5	1.0	2
PT181	0.4	0.6	0.8	1.0	2
PT186	0.2	0.3	0.4	0.5	2
ES112	0.2	0.3	0.4	0.5	2
ES120	0.2	0.3	0.4	0.5	2
ES130	0.2	0.3	0.4	0.5	2
ES418	0.2	0.3	0.4	0.5	2
ES422	0.2	0.3	0.4	0.5	2
ES615	0.2	0.3	0.4	0.5	2
ES620	0.2	0.3	0.4	0.5	2
DZ000	0.1	0.2	0.5	1.0	10

The potential increases over time to reflect planned scaling of electrolyzer deployment and export capacities. Notably, DZ000 (Algeria) has a significantly higher marginal cost ($MC = 10$ €/GWh), representing long-distance transport and less favorable cost assumptions.

These supply potentials are used to generate node-level capacity constraints through scenario-specific GAMS scripts:

- `scen_%string%_sup.gms` for supply constraints
- `lb_p(n,e,y,h) = min(lb_p(n,e,y,h), cap_p(n,e,y,h))` enforces binding limits

This mechanism enables dynamic scenario analysis while ensuring all regional supply assumptions are respected.

3.6 Arc Data and Pipeline Infrastructure

This part of the model script loads and defines all data related to pipelines (arcs), including capacities, costs, losses, repurposing, bidirectionality, and expansions. Arcs are specified between nodes and may support unidirectional or bidirectional flow of various fuels.

Parameters and Sets:

- $a_s(a,n), a_e(a,n)$: Assign start and end nodes to arc a.
- $anm(a,n,m)$: Logical map indicating existence of arc between node pairs.
- $cap_a(a,e,y)$: Volumetric capacity of arc by fuel e and year y.
- $c_a(a,e,y), c_{ab}(a,e,y)$: Transport costs, including bidirectional premiums.
- $c_{ax}(a,e,y), f_{ab}(a,y)$: Expansion and fixed bidirectional cost components.
- $c_ar(a,e,f,y), f_{ar}(a,e,f,y)$: Costs of repurposing infrastructure from fuel e to f.
- $e_a(a,e)$: Efficiency factor for arc flows.
- $is\_bid(a)$: Flag identifying bidirectional arcs.

The model loads arc definitions from the GDX file:

```
$gdxin gdx/%data%
$load a, dat_a
```

Connectivity Assignment: Start/end nodes are defined if any fuel-based length or capacity is nonzero:

```
a_s(a,n)$(sum((m,e),dat_a(a,n,m,e,'len')+dat_a(a,n,m,e,'cap'))>0)=1;
a_e(a,m)$(sum((n,e),dat_a(a,n,m,e,'len')+dat_a(a,n,m,e,'cap'))>0)=1;
anm(a,n,m)$(a_s(a,n) AND a_e(a,m))=1;
```

Bidirectional Arcs: Opposite arcs are automatically assigned for any pair with mirrored connectivity:

```
loop {(ai,ao,n,m)$(anm(ai,n,m) AND anm(ao,m,n)),
    opp(ai,ao)=1; opp(ao,ai)=1;
};
```

Defaults and Corrections: Pipeline and repurposing costs are defaulted if not explicitly provided, ensuring robust execution:

```
dat_o('BFPipe',e,'')$(...) = dat_o('BFPipe','','');   * Flow cost
dat_o('BIPipe',e,'')$(...) = dat_o('BIPipe','','');   * Investment cost
dat_o('RepurpArc',e,f)$(...) = dat_o('RepurpArc','','');
dat_o('OffshMult','','') = max(0, dat_o('OffshMult','','') - 1);
```

Cost and Efficiency Assignment: All arc cost expressions incorporate scaling by calibration factors, pipeline length, and volume conversion:

$$c_a(a,e,y) = \text{BFPipe}(e) \cdot \text{vola2}(e) \cdot \frac{(\text{len}_{\text{on}} + \text{OffshMult} \cdot \text{len}_{\text{off}}) \cdot \text{cal}_c}{\text{Len}_{\text{Std}}} \tag{8}$$

$$e_a(a,e) = 1 - \min\left(\text{LossMax}, \frac{\text{BLPipe}(e) \cdot \text{len}_{\text{on}} \cdot \text{cal}_l}{\text{Len}_{\text{Std}}}\right) \tag{9}$$

Expansion costs $c_{ax}(a,e,y)$, repurposing $c_{ar}(a,e,f,y)$, and fixed bidirectional costs $f_{ab}(a,y)$ follow a similar structure, adjusted by `YearStep`.

Offshore Constraint: To avoid invalid offshore pipe data, the following check ensures no offshore segment exceeds total arc length:

$$\text{off}_{a,n,m,e} = \min(\text{len}_{a,n,m,e}, \text{off}_{a,n,m,e}) \tag{10}$$

Calibration and Consistency: All arcs are assigned nonzero calibration parameters if undefined:

```
dat_a(a,n,m,e,'cal_c')$(...) = 1;
dat_a(a,n,m,e,'cal_l')$(...) = 1;
...
```

Symmetry in Arc Properties: If an arc a has defined properties and its opposite exists, the model ensures mirrored values are applied:

```
loop{(ai,ao,e,y)$(opp(ai,ao) AND c_a(ai,e,y)>0),
   c_a(ao,e,y) = c_a(ai,e,y); ...
};
```

This guarantees structural consistency in bidirectional pipelines, avoiding asymmetries unless explicitly modeled.

3.7 Storage Data: Capacity, Efficiency, and Cost

This section loads and processes data for gas storage facilities. The storage parameters define the capacity for injection, extraction, and working volume, as well as efficiency and associated costs.

Storage Data Loading The following GAMS command loads raw storage data from the Excel-derived GDX file:

```
$gdxin gdx/%data%
$load dat_w
```

Parameter Definitions

- `cap_wi(n,e,y)`: Injection capacity [GWh/h]
- `cap_we(n,e,y)`: Extraction capacity [GWh/h]
- `cap_ww(n,e,y)`: Working gas volume [GWh]
- `e_w(n,e)`: Storage efficiency (0–1)
- `c_we(n,e)`: Cost of extraction from storage [€/GWh]

Fallback Rules and Adjustments Missing or zero calibration values are defaulted to 1 to avoid infeasibilities:

```
dat_w(n,e,'cal_c')$(...) = 1;
dat_w(n,e,'cal_l')$(...) = 1;
```

Handling Missing Data for Specific Fuels If extraction or injection capacities are missing for hydrogen or gas, fallback estimates are inferred from the complementary fuel using conversion factors:

```
cap_we(n,e,y)$(is_h(e)) = dat_w(n,'G','X') / vols2('H');
cap_we(n,e,y)$(is_g(e)) = dat_w(n,'H','X') * vols2('H');

cap_wi(n,e,y)$(is_h(e)) = dat_w(n,'G','I') / vols2('H');
cap_wi(n,e,y)$(is_g(e)) = dat_w(n,'H','I') * vols2('H');
```

Working Gas Conversion Total storage capacity is scaled to account for representative time blocks:

$$\texttt{cap\_ww}(n,e,y) = \texttt{dat\_w}(n,e,\text{“W”}) \cdot \left(\frac{\sum_h \texttt{scaleUp}(h)}{8760}\right) \tag{11}$$

Efficiency and Extraction Cost The efficiency is defined as a percentage reduction based on a fixed loss rate:

$$\texttt{e\_w}(n, e) = 1 - 0.01 \cdot \texttt{dat\_w}(n, e, \text{“cal\_l”}) \tag{12}$$

Extraction costs are assigned based on calibrated values and scaled by storage energy conversion:

$$\texttt{c\_we}(n, e) = 1.00 \cdot \texttt{vols2}(e) \cdot \texttt{dat\_w}(n, e, \text{“cal\_c”}) \tag{13}$$

4 Input Verification for Nodes and Arcs

To ensure data consistency and model feasibility, the script includes a dedicated verification step for nodes and arcs. This check identifies:

- `abort_n(n)`: Nodes with critical data errors that would cause the model to abort.
- `warn_n(n)`: Isolated nodes with no incoming or outgoing connections.
- `abort_a`: Arcs with incomplete or infeasible input data (e.g., missing endpoints or undefined capacities).

The verification script is included using a conditional macro to enable toggling:

```
%Verify%$INCLUDE Verify_nodes_arcs.gms
```

If any of the above sets are populated during verification, the model can be configured to terminate or flag warnings, prompting the user to inspect and correct input inconsistencies. The results are exported to a separate GDX file for review:

```
execute_unload 'gdx/%string%_verify',
    abort_n,
    warn_n,
    abort_a;
```

5 Preparing Structured Output for Reporting

Once the input data has been fully loaded, validated, and transformed, the script proceeds to organize and export the structured data into a GDX file. This final step prepares the model for execution and enables efficient post-processing and reporting.

5.1 Storing Arc-Related Parameters and File Management

To facilitate scenario analysis and model output reporting, selected arc-related variables are organized into a five-dimensional reporting parameter, `rep_a`. This compact structure records key attributes for each arc in the network, including:

- `cap`: Existing arc capacity.
- `x_u`: Upper bound for capacity expansion.
- `x_l`: Lower bound for capacity expansion.
- `opp`: Capacity of the arc in the opposite direction.

```
parameter rep_a;
rep_a(a,y,n,m,e,'cap')  $(a_s(a,n) AND a_e(a,m)) = cap_a(a,e,y);
rep_a(a,y,n,m,e,'x_u')  $(a_s(a,n) AND a_e(a,m)) = ub_ax(a,y);
rep_a(a,y,n,m,e,'x_l')  $(a_s(a,n) AND a_e(a,m)) = lb_ax(a,e,y);
rep_a(a,y,n,m,e,'opp')  $(a_s(a,n) AND a_e(a,m)) = sum(ao$opp(ao,a), cap_a(ao,e,y));
```

Before saving this data, the script ensures a clean file system environment by creating the necessary folder (if it does not exist) and backing up any previous GDX files to preserve earlier versions:

```
$call if not exist "gdx" mkdir "gdx"
$call if exist "gdx/%string%_inputs_old.gdx" del "gdx/%string%_inputs_old.gdx"
$call if exist "gdx/%string%_inputs.gdx" ren "gdx/%string%_inputs.gdx" "%string%_inputs_old.gdx"
```

This mechanism is crucial for iterative development and scenario tracking.

5.2 Exporting the Final GDX File

The last operation in the script is the export of all assembled and validated input data into a single GDX file. This file is the definitive input for the model's optimization run.

```
execute_unload 'gdx/%string%_inputs',
    a,
    a_e,
    .
    .
    .
    yscai;
```

This export guarantees that all relevant model components—sets, parameters, transformations, calibration constants, and scenario values—are consolidated into a structured format, enabling reproducibility and consistency across model runs.

6 Scenario Integration: Supply and Demand

To support modular scenario analysis, the model dynamically incorporates external GAMS files that define supply and demand configurations. These scenario files are located in the `input_data` directory and follow a standardized naming convention using the model run string (e.g., `Spain_2040_4_2`).

Including Scenario Files Two scenario-specific files are loaded:

```
$INCLUDE "%path%/input_data/scen_%string%_sup.gms"
$INCLUDE "%path%/input_data/scen_%string%_dmd.gms"
```

These files should define the supply and demand assumptions used in a given model run. They typically modify node-level parameters such as production capacities, lower bounds, or demand levels for a specified case.

Expected Structure of Scenario Files While the exact contents may vary by use case, scenario files are expected to contain assignments like:

- **Supply scenarios** may override:
  - `cap_p(n,e,y,h)` – Available production capacity
  - `lb_p(n,e,y,h)` – Minimum production (e.g., for hydrogen rollout)
- **Demand scenarios** typically define:
  - `dmd(n,e,y,h)` – Node-level demand (NUTS3)
  - `dmd2(nuts2,e,y,h)` – Regional demand (NUTS2)

Notes on Use These files enable targeted experimentation across varying policy, technology, or consumption assumptions without altering the core model. Each run string corresponds to a specific scenario setup, ensuring full traceability in sensitivity analysis or planning exercises.