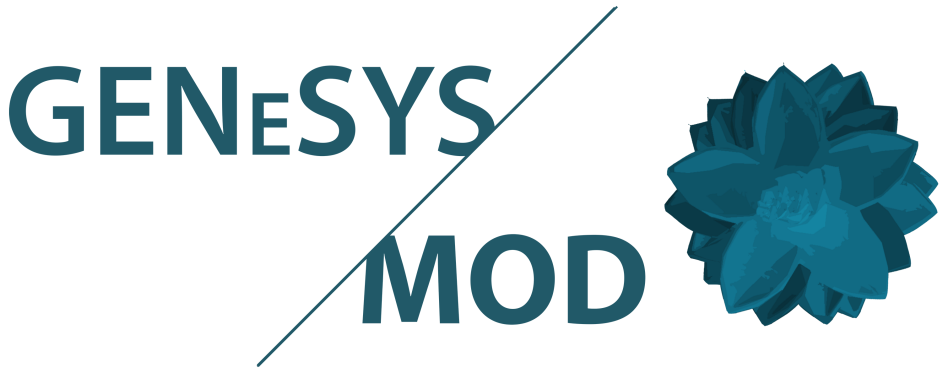

GENeSYS-MOD

The Global Energy System Model

Version 3.0

User Manual - Technical Guide

Version 1.0

December 2020

Contents

1 Prerequisites required to run GENeSYS-MOD v3.0 1

2 GAMS-files and organizational structure 1

2.1 `genesysmod.gms` 3
2.2 `genesysmod_dec.gms` 4
2.3 `genesysmod_dataload.gms` 4
2.4 `genesysmod_subsets.gms` 5
2.5 `genesysmod_timeseries_reduction.gms` and `genesysmod_timeseries_timeslices.gms` 5
2.6 `genesysmod_settings.gms` 6
2.7 `genesysmod_interpolation.gms` 6
2.8 `genesysmod_aggregate_region.gms` 6
2.9 `genesysmod_bounds.gms` 6
2.10 `genesysmod_scenariodata_[REGION].gms` 7
2.11 `genesysmod_equ.gms` 7
2.12 `genesysmod_results.gms` and `genesysmod_results_excel.gms` 8

3 Mathematical structure of GENeSYS-MOD 9

3.1 Sets 9
3.2 Parameters 11
3.2.1 Global parameters 11
3.2.2 Demand parameters 12
3.2.3 Emission parameters 12
3.2.4 Trade parameters 13
3.2.5 Reserve margin parameters 13
3.2.6 General technology parameters 14
3.2.7 Capacity and activity parameters 15
3.2.8 Storage parameters 16
3.2.9 Ramping parameters 16
3.2.10 Modal split parameters 16
3.3 List of variables 17

4 Debugging, common errors, and useful tips 26

4.1 Debugging 26
4.1.1 Debugging compilation errors 26
4.1.2 Debugging model errors 26
4.1.3 Debugging result handling errors 27
4.2 Common errors 28
4.2.1 Common compilation errors 28
4.2.2 Common model errors 30
4.3 Running batch jobs 30
4.4 Other useful tips 30

5 Support and Contact Information 31

1 Prerequisites required to run GENeSYS-MOD v3.0

1. Microsoft Windows as OS with Microsoft Excel installed[1]
2. GAMS IDE at version 30 or higher
3. Solver licenses for LP and DNLP (Recommended solvers: GUROBI and PATHNLP)
4. Usually, at least 16GB of RAM are recommended to run any detailed GENeSYS-Model. A minimum of 8GB is required for most applications. Large-scale models with high time-resolution can require upwards of 200GB of RAM - for this, a professional computation environment is needed
5. Downloaded and extracted source code of GENeSYS-MOD

For a detailed guide on how to first set up and run GENeSYS-MOD, please refer to the Quick-Start Guide, found here.

2 GAMS-files and organizational structure

In this section, the various files used in the GENeSYS-Model, as well as their interlinkages are described in more detail. The aim is to give a better understanding of the basic functions and elements of each file so that navigation around the code blocks is made easier. Small code-examples will be given from time to time. Since not all GAMS-specific terms can be explained here, we would like to refer to the GAMS user manual (found here) for help on the GAMS programming language. Certain elements might be mentioned in this section, but will not be explained in detail. A more detailed explanation of the mathematical structure can be found in section 3 instead. For general questions about the underlying nomenclature and elements of the Open-Source Energy Modeling System (OSeMOSYS), please refer to their wiki, found here.

[1] Currently, this is the only option to properly run GENeSYS-MOD, since it uses gdxxrw for data handling, which is a Windows executable. A workaround is currently planned to enable the translation between Excel and GAMS on other operating systems such as MAC OS and Linux.

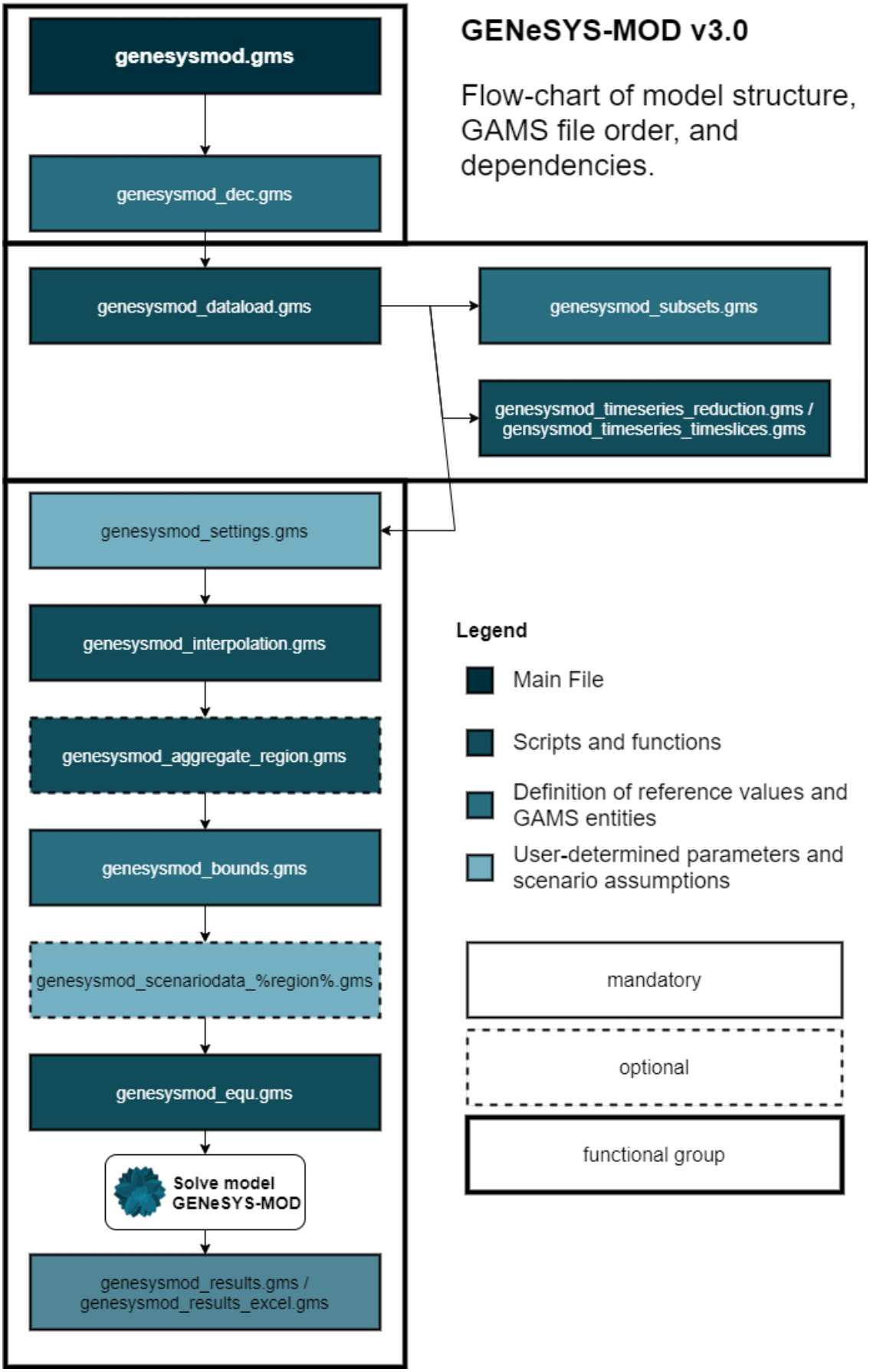

Figure 1: Overview over the primary and secondary model files used to run GENeSYS-MOD.

2.1 genesysmod.gms

The code in this file is roughly divided into five different parts:

- Default Settings
- Calling further GENeSYS-MOD-files
- Settings for Solver
- Model and Solve Statements
- Result File Creation

The default settings are specified at the beginning of the file. Within these settings, required options are specified to allow model calculations in the case where no additional options are specified with the run command. They cover a range of technical settings (e.g. location of directories, how many threads should be used, the kind of file system) but mainly set model specific options (e.g. which scenario should be calculated, the models temporal and spatial resolution, etc.). For debugging and testing purposes, a switch for only running the data loading is implemented (see example code below).

```
1 $if not set inputdir                $setglobal inputdir Inputdata\
2 $if not set threads                 $setglobal threads 2
3 $if not set model_regions           $setglobal regions europe
4 $if not set switch_test_data_load   $setglobal switch_test_data_load 1
```

After setting the default options, the different model files are being called. Some of these files require certain switches in the settings to be turned on in order to be called. Since some files build on the previous ones, it is generally not advised to change the order of the files here.

```
1 $include genesysmod_dec.gms

3 $include genesysmod_dataload.gms

5 $include genesysmod_settings.gms

7 $include genesysmod_bounds.gms

9 $include genesysmod_equ.gms
```

The next section covers some predefined options for the two commonly used solvers, CPLEX and GUROBI. As specified earlier, a license for the solver of choice is required, since the free versions of the solvers are not suitable for GENeSYS-MOD. For more information on the specific solver options, the reader is referred to the official GAMS website.

The next two statements, `model` and `solve`, build the model based on all equations and input data and

subsequently run the specified solver and solve the model. The model statement allows for a variety of options which are described in detail on the official GAMS website. The solve statement, on the other hand, is quite straight forward and has to specify the model name (assigned in the `model` statement), the class of optimization problem and the variable to optimize.
Lastly, after a succesful run of the model, the result generation file is being called which handles the entirety of result processing.

2.2 genesysmod_dec.gms

As mentioned in the introduction, the declaration file consists of all the declarations needed to run the program. There are three different types of element that can be declared: `sets`, `parameters` and `variables`. As an important note, this file only handles the declaration of the elements, while the `dataload` file is responsible for assigning the values for all the parameters.
Sets are "containers" that can be filled with any number of elements which share the same characteristics from a model point of view. Typical examples of sets in the case of GENeSYS-MOD are Region, Year, Technology, or Fuels. For an overview of all different sets, the reader is referred to Table 1. While the elements of sets can be of any kind (Technologies are words while Years are numbers), they are generally stored as strings which, in the case of years, does not allow for any calculations based on the set-elements. This issue can solved through the use of parameters (see Table 2) or specific set functions.
Parameters are values which depend on a varying number of sets. They have to be assigned before the model run by the user which in the case of GENeSYS-MOD is mostly done via the input Excel sheets.
Variables are calculated during the model run and can be of three different dimensions: positive variables, negative variables and free variables (free meaning no restriction with respect to the variables dimension). This assignment can already be done during the declaration of the variables (see example below).

```
positive variable Import(y_full,TIMESLICE,FUEL,REGION_FULL,rr_full);
negative variable Export(y_full,TIMESLICE,FUEL,REGION_FULL,rr_full);
free variable NetTrade(y_full,TIMESLICE,FUEL,REGION_FULL);
```

2.3 genesysmod_dataload.gms

In this file, the majority of the data upload is being handled. Since there are multiple Excel sheets where the data is being stored, a separate data upload routine is provided for each of these sheets.
The first block of parameters which is being loaded contains all sets and parameters which are defined globally, thus being independent from different scenarios which might be calculated. As can be seen in the code line below, a `sets` file is created on the bases of `%inputdir%%global_data_file%.xlsx` which is the global Excel data sheet in the input directory.

```
$ifi %switch_only_load_gdx%==0 $call "gdxxrw %inputdir%%global_data_file%.xlsx @%tempdir%temp_%
    global_data_file%.tmp o=%gdxdir%%global_data_file%.gdx MaxDupeErrors=999 CheckDate ";
```

The next two blocks import all the data which is scenario specific. First, sets are uploaded, since specific scenarios might include technologies or regions which are not part of all scenarios. Afterwards, a number of parameters is set for all regions, since in the global sheet they might be defined for a single `World` region. Second, scenario specific parameters are imported.

After calling `genesysmod_subsets` and a couple of parameter assignments to cover possibly missing data, depending on the time resolution defined in `genesysmod.gms`, the respective `timeseries` gams file is called.

2.4 genesysmod_subsets.gms

This file is a straightforward assignment of technologies or fuels into different subsets, since some equations or calculations require a specific subset of elements. An example is given in the code example below, where all three wind offshore technologies are grouped into the subset `Offshore`. If new set elements are added by the user, it is important to check whether they would belong into one or multiple of these subsets.

```
set Offshore(t);
Offshore(t) = no;
Offshore('Res_Wind_Offshore_Deep') = yes;
Offshore('Res_Wind_Offshore_Shallow') = yes;
Offshore('Res_Wind_Offshore_Transitional') = yes;
```

2.5 genesysmod_timeseries_reduction.gms and genesysmod_timeseries_timeslices.gms

These two files control the data handling of the hourly input files and convert the loaded data into the desired format. The hourly base data can be found and modified by editing the file set in the command line parameter `%hourly_data_file%` (e.g., Hourly_Data_REMES_v04_kl_25_05_2020).

In the case of `genesysmod_timeseries_timeslices.gms`, the capacity factors for reneweable generators as well as demand for each timeslices will be aggregated to the annual time slices, mentioned in Burandt, Löffler, and Hainsch (2018).[2]

`genesysmod_timeseries_reduction.gms` generates a more sophisticated (reduced) hourly time-series. This approach is also used by the Dynamic Investment and Dispatch Model for the Future European Electricity Market (dynELMOD), presented by Gerbaulet and Lorenz (2017).[3] In general, this time-series reduction algorithm works in three steps. First, every nth hour of a full hourly time-series is chosen, starting at a given starting-hour. If the feature is activated, an additional *"Dunkelflaute"* (a time span with little wind and solar) can be added. In this case, the first 12 or 24, depending on target resolution, consecutive hours will get a artificially reduced renewable infeed. Next, this reduced time-series is smoothed with a moving-average

[2] T. Burandt, K. Löffler, and K. Hainsch. 2018. *GENeSYS-MOD v2.0 - Enhancing the Global Energy System Model.* DIW Berlin, Data Documentation No. 95. Berlin, Germany. https://www.diw.de/documents/publikationen/73/diw_01.c.594273.de/diw_datadoc_2018-094.pdf.

[3] C. Gerbaulet and C. Lorenz. 2017. *dynELMOD: A Dynamic Investment and Dispatch Model for the Future European Electricity Market.* DIW Berlin, Data Documentation No. 88. Berlin, Germany.

function in the next step to decrease the artifacts and jumps of the new time-series. The window width is defined by hand for each technology. The third step scales the new time-series with a discontinuous non-linear program. For a detailed description of this process, please refer to Gerbaulet and Lorenz (2017).[4]

2.6 genesysmod_settings.gms

This short file handles a couple of functionalities which are not covered by the default settings in `genesysmod.gms`. Among other general inputs, the user can manually switch specific years on and off, which can help with faster testing in the initial phases of model calculations.

2.7 genesysmod_interpolation.gms

The file `genesysmod_interpolation` is required to assure data consistency for the cases where no data for specific parameters is given for a specific year. Depending on the parameter, either the previous value is taken or the average of the values before and after the year in question. An example for both cases is given in the code example below. In a future version of GENeSYS-MOD, we plan to implement the weighted average of the years surrounding the missing value, which would allow for a better representation of years which are not in the middle of two other model years. An example for both cases is given in the code example below.

```
*Value from previous period
MinStorageCharge(r,s,y)$(MinStorageCharge(r,s,y) = 0) = MinStorageCharge(r,s,y-1);

*Average value
CapitalCost(r,t,y)$(CapitalCost(r,t,y) = 0) = (CapitalCost(r,t,y-1)+CapitalCost(r,t,y+1))/2;
```

2.8 genesysmod_aggregate_region.gms

This file allows the aggregation of all model region into a single one and is thus only called, if the `%switch_aggregate_region%==1` in the main file. As a result, some parameters are either aggregated over all regions or the base-region value is taken. Some examples for both cases are given in the code example below.

```
*Sum of all regions
SpecifiedAnnualDemand('%model_region%',f,y) = sum(r,SpecifiedAnnualDemand(r,f,y));

*Base region value
OperationalLife('%model_region%',t) = OperationalLife('%data_base_region%',t);
```

2.9 genesysmod_bounds.gms

The bounds file defines different reference values and initializes defaults for parameters. This has to be done seperately, as GAMS does not allow for a default value to be set at parameter definition (as is the

[4] C. Gerbaulet and C. Lorenz. 2017. *dynELMOD: A Dynamic Investment and Dispatch Model for the Future European Electricity Market.* DIW Berlin, Data Documentation No. 88. Berlin, Germany.

case in e.g. the GMPL version of *OSeMOSYS*). Therefore, all different base values are given here, as well as some general settings, such as the definition of the *Infeasibility Technologies* (see section 4.1.2 for their usage).

```
*
* ####### Default Values #############
*

RETagTechnology(r,RES,y) = 1;
RETagFuel(r,'Power',y) = 1;
RETagFuel(r,'Heat_Low_Residential',y) = 1;

TotalAnnualMaxCapacityInvestment(REGION,TECHNOLOGY,y) = 999999;
TotalAnnualMinCapacityInvestment(REGION,TECHNOLOGY,y) = 0 ;

parameter YearlyDifferenceMultiplier(YEAR_FULL);
YearlyDifferenceMultiplier(y) = max(1,YearVal(y+1)-YearVal(y));
```

2.10 genesysmod_scenariodata_[REGION].gms

This file contains additional modifications to the data and the model that are only persistent in a specific regional analysis. For example, phase-out dates of specific energy carriers (i.e., coal phase-out in Germany in 2038) can be entered here alongside other governmental targets and goals. Also, the different scenarios and sensitivity analyses that are relevant for only one region can be entered here. The file will be dynamically loaded, if a file with the correct `%model_region%` exists in the main genesys mod folder. The scenario data file is loaded within `genesysmod.gms` and a compiler message is given, if no scenario data file can be found:

```
$ifthen exist genesysmod_scenariodata_%model_region%.gms
$include genesysmod_scenariodata_%model_region%.gms
$else
display "HINT: No scenario data for region %model_region% found!";
$endif
```

The actual contents for the different case studies and analyses differ from case to case. E.g.: Within the Chinese case study, four different scenarios are programmed here and the targets set by the current governmental five-year plan are also added as additional bounds for the model. On the other hand side, the European case study considers four completely different scenarios based on pathways created within the openENTRANCE project.

2.11 genesysmod_equ.gms

Within the `genesysmod_equ.gms` file, the actual main model equations and additional parameters are defined. First, on the top of the file, the model's objective function is declared. Next, several auxiliary parameters are being set in this file as well. These parameter control the building of the model matrix, as not each equation

is set up, if it is bounded by external parameters. Generally, the auxiliary parameters will be set based on the corresponding input parameters:

```
parameter CanFuelBeUsedInTimeslice(YEAR_FULL, TIMESLICE_FULL, FUEL, REGION_FULL);
CanFuelBeUsedInTimeslice(y,l,f,r)$
(sum((m,t), InputActivityRatio(r,t,f,m,y)*
            TotalAnnualMaxCapacity(r,t,y)*
            CapacityFactor(r,t,l,y)*
            AvailabilityFactor(r,t,y)*
            TotalTechnologyModelPeriodActivityUpperLimit(r,t)*
            TotalTechnologyAnnualActivityUpperLimit(r,t,y))
 > 0) = 1;
```

After the declaration and initialization of these auxiliary parameters, the actual equations are being listed. In GENeSYS-MOD, the declaration of the equation is always close to its initialization followed by potential additional upper and lower bounds. In the following example, the equation `EBa6_RateOfFuelUse3` is generally be declared for all years, timeslice, fuels, and regions. But the equation is actually only being set up, if the corresponding fuel can actually be produced in this timeslice, region, and year (controlled by the $-Condition in the second line). If this condition is negative, the variable that would be set in this equation is instead fixed to zero, as presented in line 3 of the snipped (otherwise it would be unbounded):

```
equation EBa6_RateOfFuelUse3(YEAR_FULL,TIMESLICE_FULL,FUEL,REGION_FULL);
EBa6_RateOfFuelUse3(y,l,f,r)$(CanFuelBeUsedInTimeslice(y,l,f,r) > 0)..
 sum(t, RateOfUseByTechnology(y,l,t,f,r)) =e= RateOfUse(y,l,f,r);
RateOfUse.fx(y,l,f,r)$(CanFuelBeUsedInTimeslice(y,l,f,r) = 0) = 0;
```

Some blocks of equations will be activated or deactivated given the global command line parameters set in the `genesysmod.gms` file. The blocks start with `$ifthen` command followed by the command line parameter and the value it has to be set to. Such a block always has to closed by a later `$endif` statement:

```
$ifthen %switch_ramping% == 1
[...]
$endif
```

2.12 genesysmod_results.gms and genesysmod_results_excel.gms

These two files handle the result processing of GENeSYS-MOD. For this purpose, a couple of parameters are defined the beginning which all have the prefix `"z_"`, indicating that this parameter contains model results (e.g. `z_TotalEnergyProduced(y,r)` contains all energy which is being produced in a region and year).
The next block maps technologies to different categories, which is useful for later aggregation and depiction of results. If new technologies are introduced, they have to be added here to the different categories if they should be considered in the respective category. The different categories are defined in the beginning of the file:

```
Set
Category / Transformation, Buildings, Industry, Storage, Resource, Power, Transportation, Area,
    CCS /
```

Afterwards, model output variables are assigned to the parameters which were defined previously. Additionally, a group of parameters is used with the prefix `"excel_"` which are then later exported into different sheets of an Excel file.

3 Mathematical structure of GENeSYS-MOD

In this section, the basic mathematical structure, as well as the parameters and variables of the model will be presented and explained. As in each mathematical optimization program, sets define the general structure of the model as these are used to define the dimension of all parameters, variables, and equations. Parameters represent for the model fixed data, whereas variables represent the decision variables that are determined by the model equations.

3.1 Sets

For defining the actual model dimensions, a large variety of sets are being used. These sets are declared in the file `genesysmod_dec.gms` and are initialized with the values from the input files (compare Section 2.3).

Table 1: Sets of GENeSYS-MOD

Name	Index	Description
YEAR	y	Represents the time-frame of the model. This set contains all years to be considered in the corresponding analysis.
TIMESLICE	l	Represents the temporal resolution within a YEAR of the analysis. This set either contains aggregated time-slices or a (reduced) consecutive hourly time-series. Each YEAR has the same amount of timeslices assigned.
REGION	r	Represents the regional scope of the model. This set can usually contains aggregated global regions, individual countries, or sub-country regions and states.
TECHNOLOGY	t	Represents the main elements of the energy system that produce, convert, or transform energy (carriers) and their proxies. Technologies can represent specific individual technology options, such as a "Natural Gas CCGT". They can also represent abstracted or aggregated collection of technologies used for accounting purposes (e.g., stock of cars).
FUEL	f	Represents energy carriers, energy services, or proxies that are consumed, produced, or transformed by TECHNOLOGIES. These can represent individual energy carriers, aggregated groups, or artificial commodities, required by the analysis to be carried out.
SECTOR	se	Represents different sectors in the energy system. Used for aggregation and accounting purposes.
EMISSION	e	Represents potential emissions that can be derived by the operation of certain TECHNOLOGIES. Typically this includes atmosphäric emissions, such as CO2.
MODE_OF_OPERATION	m	Represents the different modes in which a technologies can be operated by. A technology can have different inputs and/or outputs for each mode of operation to represent fuel-switching. E.g., a CHP can produce electricity in one mode and heat in another one.
STORAGE	s	Represents storage facilities in the model. Storages can store FUELS and are linked to specific TECHNOLOGIES.
MODALTYPE	mt	Represents a modal type (e.g., rail-transportation) used in the transportation sector of the model. These are used to control modal shifting to a certain degree.
SEASON	ls	Represents (by successive numerical values) how many seasons (e.g. winter, intermediate, summer) are accounted for and in which order. This set is needed if storage facilities are included in the model and the "short-term storages" are disabled.
DAYTYPE	ld	Represents (by successive numerical values) how many day types (e.g. workday, weekend) are accounted for and in which order. This set is needed if storage facilities are included in the model and the "short-term storages" are disabled.
DAILYTIMEBRACKET	lh	Represents (by successive numerical values) how many parts the day is split into (e.g. night, morning, afternoon, evening) and in which order these parts are sorted. This set is needed if storage facilities are included in the model and the "short-term storages" are disabled.

3.2 Parameters

The following tables contain lists of all relevant parameters. These parameter are user-defined exogenously given numerical inputs to the model. Each parameter is a function of the elements in one or more sets. For instance, *CapitalCost(r,t,y)* indicates that the capital cost is a function of the region *(r)*, the technology *(t)* and the year *(y)*.

3.2.1 Global parameters

Table 2: List of global parameters

Name	Description
YearSplit(l,y)	Duration of a modelled time slice, expressed as a fraction of the year. The sum of each entry over one modelled year should equal 1.
DiscountRate(r)	Region specific value for the discount rate, expressed in decimals (e.g. 0.05)
DaySplit(y,l,f)	Length of one DailyTimeBracket in one specific day as a fraction of the year (e.g., when distinguishing between days and night: 12h/(24h*365d)).
DaysInDayType(y,ls,ld)	Number of days for each day type, within one week (natural number, ranging from 1 to 7).
Conversionls(l,ls)	Binary parameter linking one TimeSlice to a certain Season. It has value 0 if the TimeSlice does not pertain to the specific season, 1 if it does.
Conversionld(l,ld)	Binary parameter linking one TimeSlice to a certain DayType. It has value 0 if the TimeSlice does not pertain to the specific DayType, 1 if it does.
Conversionlh(l,lh)	Binary parameter linking one TimeSlice to a certain DaylyTimeBracket. It has value 0 if the TimeSlice does not pertain to the specific DaylyTimeBracket, 1 if it does.
DepreciationMethod(r)	Parameter defining the type of depreciation to be applied. It has value 1 for sinking fund depreciation, value 2 for straight-line depreciation.
ProductionGrowthLimit(y,f)	This parameter controls the maximal increase between two years of a specific fuel production from renewable energy sources.
CurtailmentCostFactor(r,f,y)	Costs per curtailed unit of activity for certain fuels and years.
TagTechnologyToSector(t,se)	Links technologies to sectors.
YearVal(y)	Assigns each year its respective number to allow calculations based on the years.

3.2.2 Demand parameters

Name	Description
SpecifiedDemandProfile(r,f,l,y)	Annual fraction of energy-service or commodity demand that is required in each time slice. For each year, all the defined SpecifiedDemandProfile input values should sum up to 1.
SpecifiedAnnualDemand(r,f,y)	Total specified demand for a year.

3.2.3 Emission parameters

Name	Description
EmissionActivityRatio(r,t,e,m,y)	Emission factor of a technology per unit of activity, per mode of operation.
EmissionsPenaltyTagTechnology(r,t,e,y)	Activates or deactivates emission penalties for specific technologies.
EmissionContentPerFuel(f,e)	Defines the emission contents per fuel.
EmissionsPenalty(r,e,y)	Monetary penalty per unit of emission.
AnnualExogenousEmission(r,e,y)	Additional annual emissions, on top of those computed endogenously by the model.
AnnualEmissionLimit(e,y)	Annual upper limit for a specific emission generated in the whole modelled region.
AnnualSectoralEmissionLimit(e,s,y)	Annual upper limit for a specific emission generated in a certain sector for the whole modelled region.
RegionalAnnualEmissionLimit(r,e,y)	Annual upper limit for a specific emission generated in a certain modelled region.
ModelPeriodExogenousEmission(r,e)	Additional emissions over the entire modelled period, on top of those computed endogenously by the model.
ModelPeriodEmissionLimit(e)	Total model period upper limit for a specific emission generated in the whole modelled region.
RegionalModelPeriodEmissionLimit(e,r)	Total model period upper limit for a specific emission generated in a certain modelled region.
RegionalCCSLimit(r)	Total amount of storeable emissions in a certain region over the entire modelled period.

3.2.4 Trade parameters

Name	Description
TradeRoute(y,f,r,rr)	Sets the distance in km from one region to another. Also controls the ability to trade on fuel from a region to another.
TradeCosts(f,r,rr)	Costs for trading one unit of energy from one region to another.
TradeLossBetweenRegions(y,f,r,rr)	Percentage loss of traded fuel from one region to another. Used to model losses in power transmission networks.
TradeCapacity(y,f,r,rr)	Initial capacity for trading fuels from one region to another.
TradeCapacityGrowthCosts(y,f,r,rr)	Costs for adding one unit of additional trade capacity per km from one region to another.
GrowthRateTradeCapacity(y,f,r,rr)	Upper limit for adding additional trade capacities. Given as maximal percentage increase of installed capacity.
SelfSufficiency(y,f,r)	Lower bound that limits the imports of fuels in as specific year and region

3.2.5 Reserve margin parameters

Name	Description
ReserveMargin(r,y)	Minimum level of the reserve margin required to be provided for all the tagged commodities, by the tagged technologies. If no reserve margin is required, the parameter will have value 1; if, for instance, 20% reserve margin is required, the parameter will have value 1.2.
ReserveMarginTagFuel(r,f,y)	Binary parameter tagging the fuels to which the reserve margin applies. It has value 1 if the reserve margin applies to the fuel, 0 otherwise.
ReserveMarginTagTechnology(r,t,y)	Binary parameter tagging the technologies that are allowed to contribute to the reserve margin. It has value 1 for the technologies allowed, 0 otherwise.

3.2.6 General technology parameters

Name	Description
InputActivityRatio(r,t,f,m,y)	Rate of use of a fuel by a technology, as a ratio of the rate of activity. Used to express technology efficiencies.
OutputActivityRatio(r,t,f,m,y)	Rate of fuel output from a technology, as a ratio of the rate of activity.
CapitalCosts(r,t,y)	Capital investment cost of a technology, per unit of capacity.
FixedCosts(r,t,y)	Fixed O&M cost of a technology, per unit of capacity.
VariableCost(r,t,m,y)	Cost of a technology for a given mode of operation (e.g., Variable O&M cost, fuel costs, etc.), per unit of activity.
AvailabilityFactor(r,t,y)	Maximum time a technology can run in the whole year, as a fraction of the year ranging from 0 to 1. It gives the possibility to account for planned outages.
CapacityFactor(r,t,l,y)	Capacity available per each TimeSlice expressed as a fraction of the total installed capacity, with values ranging from 0 to 1. It gives the possibility to account for forced outages or variable renewable generation.
OperationalLife(r,t)	Useful lifetime of a technology, expressed in years.

3.2.7 Capacity and activity parameters

Name	Description
CapacityToActivityUnit(r,t)	Conversion factor relating the energy that would be produced when one unit of capacity is fully used in one year.
TotalAnnualMaxCapacity(r,t,y)	Total maximum existing (residual plus cumulatively installed) capacity allowed for a technology in a specified year.
TotalAnnualMinCapacity(r,t,y)	Total minimum existing (residual plus cumulatively installed) capacity allowed for a technology in a specified year.
ResidualStorageCapacity(r,t,y)	Externally given, already built, capacity of specific technologies, available in a certain year and region.
TotalAnnualMaxCapacityInvestment(r,t,y)	Maximum capacity of a technology, expressed in power units.
TotalAnnualMinCapacityInvestment(r,t,y)	Minimum capacity of a technology, expressed in power units.
TotalTechnologyAnnualActivityUpperLimit(r,t,y)	Total maximum level of activity allowed for a technology in one year.
TotalTechnologyAnnualActivityLowerLimit(r,t,y)	Total minimum level of activity allowed for a technology in one year.
TotalTechnologyModelPeriodActivityUpperLimit(r,t)	Total maximum level of activity allowed for a technology in the entire modelled period.
TotalTechnologyModelPeriodActivityLowerLimit(r,t)	Total minimum level of activity allowed for a technology in the entire modelled period.

3.2.8 Storage parameters

Name	Description
TechnologyToStorage(r,t,s,m)	Binary parameter linking a technology to the storage facility it charges. It has value 1 if the technology and the storage facility are linked, 0 otherwise.
TechnologyFromStorage(r,t,s,m)	Binary parameter linking a storage facility to the technology it feeds. It has value 1 if the technology and the storage facility are linked, 0 otherwise.
StorageLevelStart(r,s)	Level of storage at the beginning of first modelled year, in units of activity.
StorageMaxChargeRate(r,s)	Maximum charging rate for the storage, in units of activity per year.
StorageMaxDischargeRate(r,s)	Maximum discharging rate for the storage, in units of activity per year.
MinStorageCharge(r,s,y)	It sets a lower bound to the amount of energy stored, as a fraction of the maximum, with a number ranging between 0 and 1. The storage facility cannot be emptied below this level.
OperationalLifeStorage(r,s,y)	Useful lifetime of the storage facility.
CapitalCostsStorage(r,s,y)	Binary parameter linking a technology to the storage facility it charges. It has value 0 if the technology and the storage facility are not linked, 1 if they are.
ResidualStorageCapacity(r,s,y)	Binary parameter linking a storage facility to the technology it feeds. It has value 0 if the technology and the storage facility are not linked, 1 if they are.

3.2.9 Ramping parameters

Name	Description
RampingUpFactor(r,t,y)	Defines how much of the built capacity can be activated each timeslice.
RampingDownFactor(r,t,y)	Defines how much of the built capacity can be deactivated each timeslice.
ProductionChangeCost(r,t,y)	Cost per unit of activated or deactivated capacity per timeslice.
MinActiveProductionPerTimeslice(y,l,f,t,r)	Minimum fuel production from specific technologies in a certain timeslice. Represents minimum active capacity requirements.

3.2.10 Modal split parameters

Name	Description
TagTechnologyToModalType(t,m,mt)	Links technology production by mode of operation to modal stype.
ModalSplitByFuelAndModalType(r,f,y,mt)	Lower bound of production of certain fuels by specific modal types.

3.3 List of variables

Table 3: List of Variables

Name	Domain	Description
NewCapacity(y,t,r)	$\mathbb{R}_{\geq 0}$	Newly installed capacity of technology t in year y.
AccumulatedNewCapacity(y,t,r)	$\mathbb{R}_{\geq 0}$	Cumulative newly installed capacity of technology t from the beginning of the time domain to year y.
TotalCapacityAnnual(y,t,r)	$\mathbb{R}_{\geq 0}$	Total existing capacity of technology t in year y (sum of cumulative newly installed and pre-existing capacity).
RateOfActivity(y,l,t,m,r)	$\mathbb{R}_{\geq 0}$	Intermediate variable. It represents the activity of technology t in one mode of operation and in time slice l, if the latter lasted the whole year.
RateOfTotalActivity(y,l,t,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfActivity of a technology over the modes of operation.
TotalTechnologyAnnualActivity(y,t,r)	$\mathbb{R}_{\geq 0}$	Total annual activity of technology t.
TotalAnnualTechnologyActivityByMode(y,t,m,r)	$\mathbb{R}_{\geq 0}$	Annual activity of technology t in mode of operation m.
RateOfProductionByTechnologyByMode(y,l,t,m,f,r)	$\mathbb{R}_{\geq 0}$	Intermediate variable. It represents the quantity of fuel f that technology t would produce in one mode of operation and in time slice l, if the latter lasted the whole year. It is a function of the variable RateOfActivity and the parameter OutputActivityRatio.
RateOfProductionByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfProductionByTechnologyByMode over the modes of operation.
ProductionByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Production of fuel f by technology t in time slice l.
ProductionByTechnologyAnnual(y,t,f,r)	$\mathbb{R}_{\geq 0}$	Annual production of fuel f by technology t.
RateOfProduction(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfProductionByTechnology over all the technologies.

Name	Domain	Description
NewCapacity(y,t,r)	$\mathbb{R}_{\geq 0}$	Newly installed capacity of technology t in year y.
AccumulatedNewCapacity(y,t,r)	$\mathbb{R}_{\geq 0}$	Cumulative newly installed capacity of technology t from the beginning of the time domain to year y.
TotalCapacityAnnual(y,t,r)	$\mathbb{R}_{\geq 0}$	Total existing capacity of technology t in year y (sum of cumulative newly installed and pre-existing capacity).
RateOfActivity(y,l,t,m,r)	$\mathbb{R}_{\geq 0}$	Intermediate variable. It represents the activity of technology t in one mode of operation and in time slice l, if the latter lasted the whole year.
RateOfTotalActivity(y,l,t,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfActivity of a technology over the modes of operation.
TotalTechnologyAnnualActivity(y,t,r)	$\mathbb{R}_{\geq 0}$	Total annual activity of technology t.
TotalAnnualTechnologyActivityByMode(y,t,m,r)	$\mathbb{R}_{\geq 0}$	Annual activity of technology t in mode of operation m.
RateOfProductionByTechnologyByMode(y,l,t,m,f,r)	$\mathbb{R}_{\geq 0}$	Intermediate variable. It represents the quantity of fuel f that technology t would produce in one mode of operation and in time slice l, if the latter lasted the whole year. It is a function of the variable RateOfActivity and the parameter OutputActivityRatio.
RateOfProductionByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfProductionByTechnologyByMode over the modes of operation.
ProductionByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Production of fuel f by technology t in time slice l.
ProductionByTechnologyAnnual(y,t,f,r)	$\mathbb{R}_{\geq 0}$	Annual production of fuel f by technology t.
RateOfProduction(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfProductionByTechnology over all the technologies.
Production(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Total production of fuel f in time slice l. It is the sum of the ProductionByTechnology over all technologies.

Name	Domain	Description
RateOfUseByTechnologyByMode(y,l,t,m,f,r)	$\mathbb{R}_{\geq 0}$	Intermediate variable. It represents the quantity of fuel f that technology t would use in one mode of operation and in time slice l, if the latter lasted the whole year. It is the function of the variable RateOfActivity and the parameter InputActivityRatio.
RateOfUseByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the RateOfUseByTechnologyByMode over the modes of operation.
UseByTechnologyAnnual(y,t,f,r)	$\mathbb{R}_{\geq 0}$	Annual use of fuel f by technology t.
RateOfUse(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Sum of the UseByTechnologyAnnual over all the technologies.
UseByTechnology(y,l,t,f,r)	$\mathbb{R}_{\geq 0}$	Use of fuel f by technology t in time slice l.
Use(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Total use of fuel f in time slice l. It is the sum of the UseByTechnology over all technologies.
ProductionAnnual(y,f,r)	$\mathbb{R}_{\geq 0}$	Total annual production of fuel f. It is the sum of the variable Production over all technologies.
UseAnnual(y,f,r)	$\mathbb{R}_{\geq 0}$	Total annual use of fuel f. It is the sum of the variable Use over all technologies.
Curtailment(y,l,f,r)	$\mathbb{R}_{\geq 0}$	Curtailment of a specifc fuel
CurtailmentAnnual(y,f,r)	$\mathbb{R}_{\geq 0}$	Sum of curtailment in all timeslices
DispatchDummy(r,l,t,y)	$\mathbb{R}_{\geq 0}$	Slack variable for dispatchable technologies for allowing the reduction of their activity
CapitalInvestment(y,t,r)	$\mathbb{R}_{\geq 0}$	Undiscounted investment in new capacity of technology t. It is a function of the NewCapacity and the parameter CapitalCost.
DiscountedCapitalInvestment(y,t,r)	$\mathbb{R}_{\geq 0}$	Investment in new capacity of technology t, discounted through the parameter DiscountRate.
SalvageValue(y,t,r)	$\mathbb{R}_{\geq 0}$	Salvage value of technology t in year y, as a function of the parameters OperationalLife and DepreciationMethod.

Name	Domain	Description
DiscountedSalvageValue(y,t,r)	$\mathbb{R}_{\geq 0}$	Salvage value of technology t, discounted through the parameter DiscountRate.
OperatingCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Undiscounted sum of the annual variable and fixed operating costs of technology t.
DiscountedOperatingCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Annual OperatingCost of technology t, discounted through the parameter DiscountRate.
AnnualVariableOperatingCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Annual variable operating cost of technology t. Derived from the TotalAnnualTechnologyActivityByMode and the parameter VariableCost.
AnnualFixedOperatingCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Annual fixed operating cost of technology t. Derived from the TotalCapacityAnnual and the parameter FixedCost.
TotalDiscountedCost(y,r)	$\mathbb{R}_{\geq 0}$	Sum of the TotalDiscountedCostByTechnology over all the technologies.
TotalDiscountedCostByTechnology(y,t,r)	$\mathbb{R}_{\geq 0}$	Difference between the sum of discounted operating cost / capital cost / emission penalties and the salvage value.
ModelPeriodCostByRegion(r)	$\mathbb{R}_{\geq 0}$	Sum of the TotalDiscountedCost over all modelled years.
AnnualCurtailmentCost(y,f,r)	$\mathbb{R}_{\geq 0}$	Undiscounted annual costs for curtailing specifc fuels
DiscountedAnnualCurtailmentCost(y,f,r)	$\mathbb{R}_{\geq 0}$	Annual costs for curtailing specifc fuels, discounted through the parameter DiscountRate.
RateOfStorageCharge(s,y,ls,ld,lh,r)	$\mathbb{R}$	Intermediate variable. It represents the commodity that would be charged to the storage facility s in one time slice if the latter lasted the whole year. It is a function of the RateOfActivity and the parameter TechnologyToStorage.

Name	Domain	Description
RateOfStorageDischarge(s,y,ls,ld,lh,r)	$\mathbb{R}$	Intermediate variable. It represents the commodity that would be discharged from storage facility s in one time slice if the latter lasted the whole year. It is a function of the RateOfActivity and the parameter TechnologyFromStorage.
NetChargeWithinYear(s,y,ls,ld,lh,r)	$\mathbb{R}$	Net quantity of commodity charged to storage facility s in year y. It is a function of the RateOfStorageCharge and the RateOfStorageDischarge and it can be negative.
NetChargeWithinDay(s,y,ls,ld,lh,r)	$\mathbb{R}$	Net quantity of commodity charged to storage facility s in daytype ld. It is a function of the RateOfStorageCharge and the RateOfStorageDischarge and can be negative.
StorageLevelYearStart(s,y,r)	$\mathbb{R}_{\geq 0}$	Level of stored commodity in storage facility s in the first time step of year y.
StorageLevelYearFinish(s,y,r)	$\mathbb{R}_{\geq 0}$	Level of stored commodity in storage facility s in the last time step of year y.
StorageLevelSeasonStart(s,y,ls,r)	$\mathbb{R}_{\geq 0}$	Level of stored commodity in storage facility s in the first time step of season ls.
StorageLevelDayTypeStart(s,y,ls,ld,r)	$\mathbb{R}_{\geq 0}$	Level of stored commodity in storage facility s in the first time step of daytype ld.
StorageLevelDayTypeFinish(s,y,ls,ld,r)	$\mathbb{R}_{\geq 0}$	Level of stored commodity in storage facility s in the last time step of daytype ld.
StorageLowerLimit(s,y,r)	$\mathbb{R}_{\geq 0}$	Minimum allowed level of stored commodity in storage facility s, as a function of the storage capacity and the user-defined MinStorageCharge ratio.

Name	Domain	Description
StorageUpperLimit(s,y,r)	$\mathbb{R}_{\geq 0}$	Maximum allowed level of stored commodity in storage facility s. It corresponds to the total existing capacity of storage facility s (summing newly installed and pre-existing capacities).
AccumulatedNewStorageCapacity(s,y,r)	$\mathbb{R}_{\geq 0}$	Cumulative capacity of newly installed storage from the beginning of the time domain to year y.
NewStorageCapacity(s,y,r)	$\mathbb{R}_{\geq 0}$	Capacity of newly installed storage in year y.
CapitalInvestmentStorage(s,y,r)	$\mathbb{R}_{\geq 0}$	Undiscounted investment in new capacity for storage facility s. Derived from the NewStorageCapacity and the parameter CapitalCostStorage.
DiscountedCapitalInvestmentStorage(s,y,r)	$\mathbb{R}_{\geq 0}$	Investment in new capacity for storage facility s, discounted through the parameter DiscountRate.
SalvageValueStorage(s,y,r)	$\mathbb{R}_{\geq 0}$	Undiscounted salvage value of storage facility s in year y, as a function of the parameters OperationalLifeStorage and DepreciationMethod.
DiscountedSalvageValueStorage(s,y,r)	$\mathbb{R}_{\geq 0}$	Salvage value of storage facility s, discounted through the parameter DiscountRate.
TotalDiscountedStorageCost(s,y,r)	$\mathbb{R}_{\geq 0}$	Difference between the discounted capital investment in new storage facilities and the salvage value in year y.
TotalActivityInReserveMargin(r,y,l)	$\mathbb{R}_{\geq 0}$	Total activity of the technologies required to provide reserve margin.
DemandNeedingReserveMargin(y,l,r)	$\mathbb{R}_{\geq 0}$	Quantity of fuel produced that is assigned to a target of reserve margin. Derived from the RateOfProduction and the parameter ReserveMarginTagFuel.

Name	Domain	Description
TotalREProductionAnnual(y,r,f)	$\mathbb{R}$	Annual production by all technologies tagged as renewable in the model. Derived from the ProductionByTechnologyAnnual and the parameter RETagTechnology.
RETotalDemandOfTargetFuelAnnual(y,r,f)	$\mathbb{R}$	Annual production of fuels tagged as renewable in the model. Derived from the RateOfProduction and the parameter RETagFuel.
TotalTechnologyModelPeriodActivity(t,r)	$\mathbb{R}$	Sum over all years for TotalTechnologyAnnualActivity.
AnnualTechnologyEmissionByMode(y,t,e,m,r)	$\mathbb{R}$	Annual emission of agent e by technology t in mode of operation m. Derived from the RateOfActivity and the parameter EmissionActivityRatio.
AnnualTechnologyEmission(y,t,e,r)	$\mathbb{R}$	Sum of the AnnualTechnologyEmissionByMode over the modes of operation.
AnnualTechnologyEmissionPenaltyByEmission(y,t,e,r)	$\mathbb{R}$	Undiscounted annual cost of emission e by technology t. It is a function of the AnnualTechnologyEmission and the parameter EmissionPenalty.
AnnualTechnologyEmissionsPenalty(y,t,r)	$\mathbb{R}$	Total undiscounted annual cost of all emissions generated by technology t. Sum of the AnnualTechnologyEmissionPenaltyByEmission over all the emitted agents.
DiscountedTechnologyEmissionsPenalty(y,t,r)	$\mathbb{R}$	Annual cost of emissions by technology t, discounted through the DiscountRate.
AnnualEmissions(y,e,r)	$\mathbb{R}$	Sum of the AnnualTechnologyEmission over all technologies.
ModelPeriodEmissions(e,r)	$\mathbb{R}$	Total system emissions of agent e in the model period, accounting for both the emissions by technologies and the user defined ModelPeriodExogenousEmission.
WeightedAnnualEmissions(y,e,r)	$\mathbb{R}$	Sum of the AnnualTechnologyEmission over all technologies weighted by the difference between two years.

Name	Domain	Description
AnnualSectoralEmissions(y,e,se,r)	$\mathbb{R}$	Sum of the AnnualTechnologyEmission over all technologies in a certain sector.
Import(y,l,f,r,rr)	$\mathbb{R}_{\geq 0}$	Imported fuel from region rr to region r in timeslice l.
Export(y,l,f,r,rr)	$\mathbb{R}_{\geq 0}$	Exported fuel from region r to region rr in timeslice l.
NewTradeCapacity(y,f,r,rr)	$\mathbb{R}_{\geq 0}$	Additional trade capacity for transmission of fuel f between regions r and rr in year y.
TotalTradeCapacity(y,f,r,rr)	$\mathbb{R}_{\geq 0}$	Sum of endogenous variable NewTradeCapacity and exogenous parameter TradeCapacity.
NewTradeCapacityCosts(y,f,r,rr)	$\mathbb{R}_{\geq 0}$	Undiscounted costs for newly installed transmission capacity.
DiscountedNewTradeCapacityCosts(y,f,r,rr)	$\mathbb{R}_{\geq 0}$	Costs for newly installed transmission capacity, discounted through the parameter DiscountRate.
NetTrade(y,l,f,r)	$\mathbb{R}$	Difference between Export and import from region r to region rr in timeslice l.
NetTradeAnnual(y,f,r)	$\mathbb{R}$	Sum over NetTrade in all timeslices in year y.
TotalTradeCosts(y,l,r)	$\mathbb{R}$	Undiscounted costs for trading fuel f from regions r to rr in timeslice l.
AnnualTotalTradeCosts(y,r)	$\mathbb{R}$	Undiscounted annual costs for trading fuel f from regions r to rr.
DiscountedAnnualTotalTradeCosts(y,r)	$\mathbb{R}$	Annual costs for trading fuel f from regions r to rr, discounted through the parameter DiscountRate.
DemandSplitByModalType(MODALTYPE,l,r,f,y)	$\mathbb{R}_{\geq 0}$	Share of demand for fuel f in region r, timeslice l, and year y that has to be provided by modaltype mt.
ProductionSplitByModalType(MODALTYPE,l,r,f,y)	$\mathbb{R}_{\geq 0}$	Share of production of fuel f in region r, timeslice l, and year y that will be provided by modaltype mt.
ProductionUpChangeInTimeslice(y,l,f,t,r)	$\mathbb{R}_{\geq 0}$	Upwards change in production between timeslice l and the previous timeslice.

Name	Domain	Description
ProductionDownChangeInTimeslice(y,l,f,t,r)	$\mathbb{R}_{\geq 0}$	Downwards change in production between timeslice l and the previous timeslice.
AnnualProductionChangeCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Undiscounted annual costs for changing the production between two timeslices.
DiscountedAnnualProductionChangeCost(y,t,r)	$\mathbb{R}_{\geq 0}$	Annual costs for changing the production between two timeslices, discounted through the DiscountRate.

4 Debugging, common errors, and useful tips

4.1 Debugging

When debugging, always check your GAMS output for errors. The GAMS process windows will give you all necessary information on the problems in your model run. Errors can be classified into three categories: (i) errors while compiling the GAMS code (thus before the model run), (ii) errors within the model runs, e.g. due to an infeasible model, and (iii) errors in the result handling process.

The model run should usually always terminate with an Exit Code of 0, stating "Normal Completion".

```
--- genesysmod.gms(3264) 272 Mb
--- GDX File C:\Users\testbed\Documents\genesys-mod\test2.gdx
*** Status: Normal completion
--- Job genesysmod.gms Stop 12/30/20 10:26:02 elapsed 0:00:10.050
```

DISCLAIMER: Normal completion means that the GAMS code ran without any issues and that a mathematically optimal solution for the optimization problem has been found. This does, however, clearly depend on the input data, scenario assumptions and constraints that you defined and that there could still be errors in these. Therefore, extensive result analysis is always required.

4.1.1 Debugging compilation errors

Compilation errors will usually be highlighted in bright red in the GAMS status window. An example of such an error message is shown below. Compilation errors usually come from mistakes in the GAMS formulation and syntax, but might also be due to missing data.

```
--- .genesysmod_bounds.gms(36) 111 Mb 1 Error
*** Error 171 in C:\Users\chimera\Documents\genesys-mod\genesysmod_bounds.gms
    Domain violation for set
--- .genesysmod_bounds.gms(265) 111 Mb
```

Double clicking these error messages leads you to the point where GAMS places the problem. Always start with the upmost error message and work downwards. Usually, fixing one error can resolve multiple issues at once. Try to re-run the compilation after each fixed error to make sure that it worked. Common compilation errors are pointed out in subsection 4.2.

4.1.2 Debugging model errors

Model errors arise when the solver is not able to find an optimal solution to your model setup. This can have various reasons: the model might actually be infeasible (e.g. due to some wrong data entries or settings), or the solver might just have difficulties finding the model solution. If the solver determines that the model is infeasible, it will display an error message:

```
Solved in 0 iterations and 4.46 seconds
Infeasible or unbounded model
LP status(4): Model was proven to be either infeasible or unbounded.
***
*** Gurobi reports the model to be either infeasible or unbounded.
*** Use option 'rerun 1' in a GAMS/Gurobi option file
*** to determine the primal solution status.
***
```

If your model is infeasible, this usually means that some constraints contradict each other (e.g. reaching zero emissions, but having a must-run constraint for a coal-based power plant). This can happen due to missing data, or just because of some data assumptions that do not match.
For this, some special technologies called *infeasibility technologies* have been introduced. They are enabled via the switch *switch_infeasibility_tech* in the *genesysmod.gms* file and can produce their respective fuels without any energy input, but at a prohibitive price. If enabled, they will show you which regions cannot produce a certain fuel and guide you towards the problems in your data or constraints.
If the model is still returning infeasible after you enable these *infeasibility technologies*, then you will have to check which equation is infeasible. To do so, check the listing (.lst) file that is generated with your GAMS run (thus regularly called *genesysmod.lst*) and search for the term "infes". This will lead you to the equation or variable that the solver determined to be infeasible.

If your solver takes an unusually long time to perform the crossover (the computational step after the barrier, usually highlighted via the solver output), you might have some issues with your data or constraints. They might not *quite* make your model infeasible, but make the optimization process for the solver extremely challenging. A good first hint for such problems might be a message stating "Sub-Optimal Termination" in your barrier when using **GUROBI**, or a "*" next to your last barrier iteration when using **CPLEX**. In case of such a seemingly "infinite loop", try to check your data points and go over your last changes.
An issue that might cause such problems might be the existence of very large discrepancies within your model data, since the solvers usually have trouble comparing very small numbers and very large numbers. You can check this in the matrix range statistics that the solver provides:

```
Starting Gurobi...
Optimize a model with 4342452 rows, 5515467 columns and 13675545 nonzeros
Coefficient statistics:
  Matrix range     [1e-03, 3e+06]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e-06, 1e+06]
Presolve removed 1613777 rows and 2237885 columns
Presolve time: 4.46s
```

GUROBI might also issue a warning if it detects scaling problems with your data.

4.1.3 Debugging result handling errors

Result handling errors occur after the model has been solved, within the result-handling gams files. This usually results in the message "Execution Errors" in GAMS at the end of the process, which is the same message that happens when the model is returned infeasible. It is thus easily confused with an infeasible model, but does not necessarily mean that. One should always check the solution report of the solver to check for an optimal solution first! If the model reports "LP status(4): Model was proven to be either infeasible or unbounded.", then refer to the previous subsection on debugging model errors.

```
--- genesysmod.gms(4401) 8435 Mb
*** Error at line 4401: division by zero (0)
--- genesysmod.gms(4453) 8435 Mb 1 Error
*** Error at line 4453: division by zero (0)
--- genesysmod.gms(4454) 8435 Mb 2 Errors
```

Result-handling errors are usually highlighted in blue and contain the message "division by zero". If your model did solve optimally, this might be caused by some calculations in the results file, if certain technologies are not used. These errors should be rather rare and one should always first check for issues within the model.

4.2 Common errors

4.2.1 Common compilation errors

Since compilation errors usually stem from a wrong usage of the GAMS programming language, we want to refer to the GAMS user manual (found here) for any errors you might encounter. However, some of the most common encountered problems are listed below:

Problem	Description	Solution
gdxxrw: Input file not found, followed by lots of errors	This error usually is a bit tricky to spot for new users, as it is not highlighted in red like other GAMS errors. It is just written in regular black, found at the beginning of the output log. This error usually comes with a large number of other GAMS-related errors and is therefore often overlooked.	Check the input file name you specified and that it is correctly located in the InputData directory.
gdxin: Cannot read gdx file. File is not a valid gdx file	This error can occur from time to time, usually after a solve has been manually aborted with the stop command, while data loading was in progress. GAMS has created an empty gdx file and therefore cannot read from it.	Go to your GdxFiles directory and delete any input gdx files that are present. GAMS should now reconstruct these files with the new run.
GAMS: Domain violation for element	This error occurs when you specify an element of a set that is not present in the data. Usually this is due to typos or wrong placement of indices (e.g. OperationalLife('RES_Hydro_Large',r) instead of OperationalLife(r,'RES_Hydro_Large')).	Check the indices and make sure that all elements you refer to are indeed included in your sets. For this, it might be good to check all input gdx and/or perform a test run with the switch "test_data_load" enabled.
GAMS: Dimension different - The symbol is referenced with more/less indices as declared	This error occurs when you have a mistake in the indices of your parameter or variable. If the parameter is declared with the sets region and year, both of these have to be present and in the correct order.	Make sure that you use the parameters and variables exactly as specified in the declaration file.
GAMS: Assigned set used as domain	This error occurs when defining a new parameter or variable. GAMS distinguished between fixed and dynamic sets.	For new parameters and variables that you define, you need to use these fixed sets for the defition. In the case of e.g. the set year or y, this would be year_full or y_full. These declarations can be found in the genesysmod.dec file.
GAMS: Solve statement not checked because of previous errors	This error usually comes in combination with other errors and only acts as an intermediate warning.	Scroll upwards to find the first error that is listed. This error does not always need to be highlighted in red! Also check for other errors and warnings in the log.
GAMS: Symbol declared but no values have been assigned	This error usually comes in combination with other errors, as GAMS will stop the data handling process as soon as is encounters any compilation errors. If this error, however, appears by itself without any other previous errors, this means that you are missing some essential data for parameters. E.g. an upper limit that needs to be defined for the model to be able to run.	Check your input data carefully. You can use the function "switch_test_dataload" to check that all input data is correctly read to GAMS. Also check for any changes that you have made to parameter definitions and equations.

4.2.2 Common model errors

Common model errors include:

1. Model has been proven infeasible.
2. Sub-optimal termination in the barrier.
3. Crossover in "infinite loop".

All of these errors point to problems in your data or defined constraints. The correct debugging procedure has been described in section 4.1.2. Usually, if the solver immediately determines the model to be infeasible, the error is with a "hard limit", such as a constraint being set too strict (or an upper limit not being correctly read from the data). If the barrier termination is sub-optimal or if the crossover takes a very long time, this can usually point to scaling problems in your data, or very tight constraints (where the model is just barely feasible).

4.3 Running batch jobs

In some instances, running the model via the command line might be preferred (e.g. to parallelize multiple runs or to automate some model variations). This can be achieved via standard GAMS functionality, using the command *gams* in a Windows command line interface, or a Windows Batch File (.bat). The usage of *gams* behaves the same as the command line within the GAMS IDE - you can set all options and command-line variables.

This means that all switches and options that you find in the *genesysmod-gms* file can be put in a batch command. An example for such a Batch command would be:

```
gams genesysmod.gms –gdxCompress=1 ––emissionPathway=DirectedVision ––emissionspenalty=1000 –gdx=
    DirectedVision_1000_488 ––elmod_nthhour=488 ––elmod_hour_steps=4 ––threads=6 –o=BatchFiles\
    Logs\DV_1000_488.log ––Info=DV_1000_488
```

Usually, Batch Files for use with GENeSYS-MOD are stored in the *BatchFiles* subfolder.

4.4 Other useful tips

In some instances, changing the solver options from their recommended (and pre-defined) values can prove beneficial for some users. Specifically the LP solver options can change computation time quite drastically. Please refer to the user manuals of the used LP solver for additional options.

Always generate a full gdx file when running the model (option -gdx=name in the command line). This file contains all information available, including marginals of constraints, all variable values and bounds, as well as all input data. This information should always be checked first when looking for errors, especially to compare it to input data from Excel to check for data transmission errors.

You can either set a fixed amount of threads for the solver to use when computing the model results, or an offset. The option *threads* in the *genesysmod.gms* file allows for these settings. Setting threads to 0 will use all available cores & threads of your machine, while setting it to e.g. -2 will use all but two available threads.

This is useful when computing your model on multiple machines with different core counts. Incorrect settings (e.g. by setting the *threads* variable to more logical cores than your machine possesses), will reset the amount of used threads to 1.

When running multiple instances of the same scenario, use the `Info` variable of GAMS to properly distinguish between all model runs. The text in `Info` will be appended to all output files.

When the model is finished after few seconds, without starting to solve the model and any obvious errors, check the command-line parameter *switch_test_data_load* in the *genesysmod.gms* file. If this parameter is active, only the data will be loaded, but no actual solve of the model is being started.

5 Support and Contact Information

For any enquiries regarding open questions and support, please contact genesysmod@coaltransitions.org or visit our GitLab page at https://git.tu-berlin.de/genesysmod/genesys-mod-public.