GENeSYS-MOD - Run Guide

Table of contents

Table of contents.......... 2
Introduction.......... 3
1. Getting ready with GENeSYS-MOD.......... 4
1.1 Getting the newest GENeSYS-MOD Version.......... 4
1.2 Setting-up VSCode.......... 4
1.3 Switching Branches in SourceTree.......... 5
1.4 Updating Subsets in Input Data file.......... 7
2. How to run GENeSYS-MOD.......... 8
2.4 The run-file.......... 8
3. Calibrating the Model for the Baseyear & 2025.......... 11
3.1.1 Disabling the No New Capacity Constraint for the BaseYear.......... 11
3.1.2 Calibrating the Power Sector in the BaseYear.......... 12
3.1.3 Base Year Bounds Debugging Power Sector.......... 12
3.1.4 Manual Residual Capacity for Heating and Transport.......... 12
3.2 Base Year Bounds Debugging.......... 13
3.2.1 What does switch_base_year_bounds_debugging do?.......... 13
3.2.2 What do BaseYearBounds_TooLow and TooHigh do?.......... 13
3.3 Calibrating 2025 & other Years.......... 15
4. Dealing with errors.......... 16
5. Visualization.......... 17
5.1 Get Tableau licences.......... 17
3.2 Merging Result Files using merge.bat.......... 18
5.3 How to use Tableau.......... 19

Introduction

The following document provides additional information on how to run the Global Energy System model (GENeSYS-MOD), how to debug the model, and how to visualize results using Tableau.

This Guide provides additional assistance and answers some common questions for users. If you have a question or advice on what should be added, feel free to comment in the document. We hope this guide will grow as more people use the model.

For further information please go to the main Folder with further information on the model and the Installation and Data Guides:
https://drive.google.com/drive/folders/1NQN1wFiDozy7L8Vo1x_gK06aIjarRZQ3

Contact: joh@wip.tu-berlin.de

1. Getting ready with GENeSYS-MOD

Before running GENeSYS-MOD and to ensure you're always using the latest version, please follow the steps below. Some of these only need to be done once; others should be repeated periodically to keep your installation up to date.

1.1 Getting the newest GENeSYS-MOD Version

If you have previously used the training material for OM4A, you will not have the newest version of GENeSYS-MOD on your machine. When doing your own country case study, you will want to use the updates that increase usability and performance. To do so, we advise you to do a fresh install, following the installation Guide. When you clone the repository, this will automatically get you the newest stable GENeSYS-MOD version. If we have prepared a branch for your case-study, follow section 1.3 on how to switch to a branch.

After switching to the new Branch, you can simply pull updates when available by going to SourceTree and Fetching & Pulling the newest updates. Should you have changes that have not been committed and are not supposed to be committed, you have to discard them before pulling the newest version.

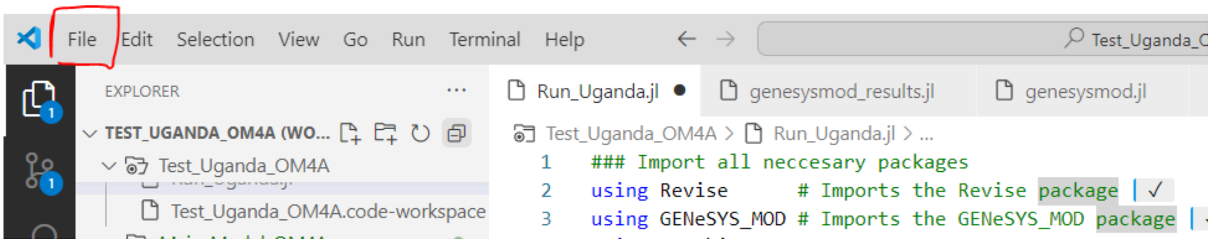
NOTE!
If you want to use the training material with the example data etc., you will need to switch to the OM4A_LOTR_Training branch. Follow Section 1.3 to see how to switch branches.

1.2 Setting-up VSCode

As explained in the installation Guide, you will have the folder containing the run-file, the input data folder, and results folder open in VSCode and will have created an environment from that folder (see installation Guide).

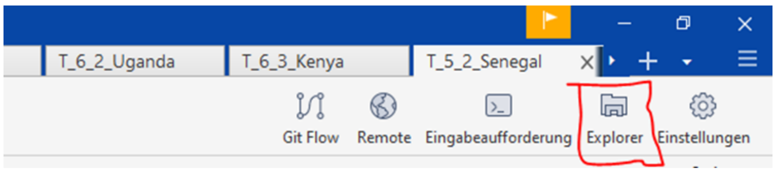
To ensure you also have access to the Julia code for GENeSYS-MOD via VSCode you will have to add the folder that contains all GENeSYS-MOD files as pulled from GitHub via Sourcetree.

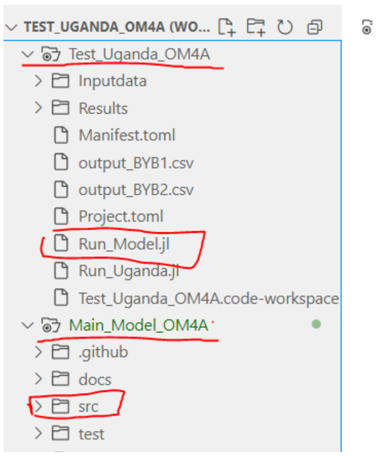
To add the folder, simply press on File (see image) and choose "Add Folder to workspace".



```
### Import all neccesary packages
using Revise        # Imports the Revise package | ✓
using GENeSYS_MOD # Imports the GENeSYS_MOD package |
```

The folder is in the location where you put it when cloning the repository from SourceTree. If you are not sure, where that is, you can go to SourceTree and press on "explorer" on the top right side of the page, this will open the folder containing the files and allow you to find the correct path.



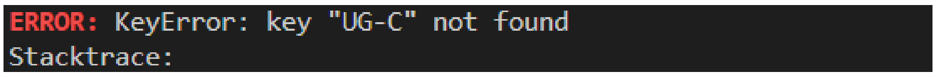


You should now have the folder containing the run-file and the folder containing the model code (src) to easily access on the left-hand side in VSCode

To finish the set-up, safe the wokspace by clicking on File and using “Save Workspace as”. For easy access, its best to put the workspace file in the same folder as the run-file.

1.2 Defining the correct base-region in utils.jl

If the model gives an error after running the first time and you get the following error:



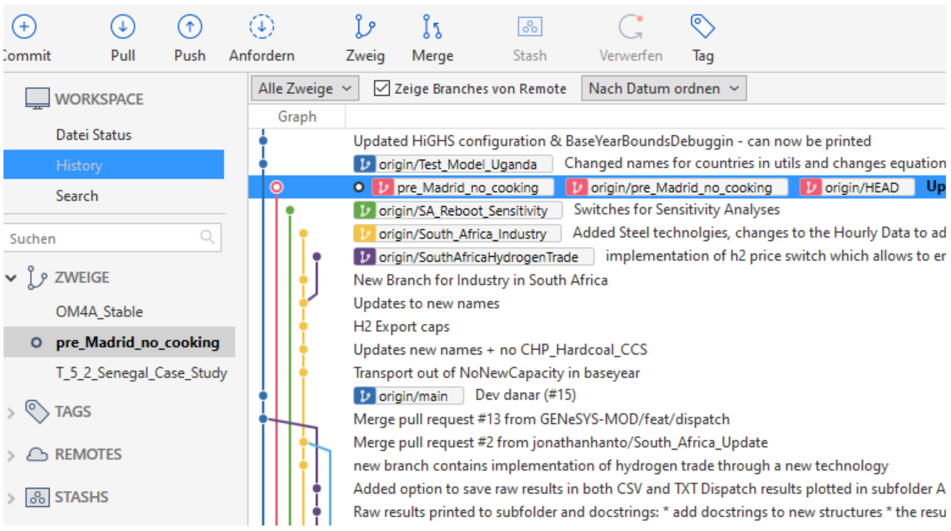
```
ERROR: KeyError: key "UG-C" not found
Stacktrace:
```

The key might differ. You then have to access the utils.jl in the src folder, go to line 69:
“`function create_daa(in_data::XLSX.XLSXFile, tab_name, els...;inherit_base_world=false,copy_world=false, base_region="UG-C")`”

You then have to change the base_region to the base-region you are using in the data. Make sure you save the changes and then reload GENeSYS-MOD to ensure the changes are used in the next model run.

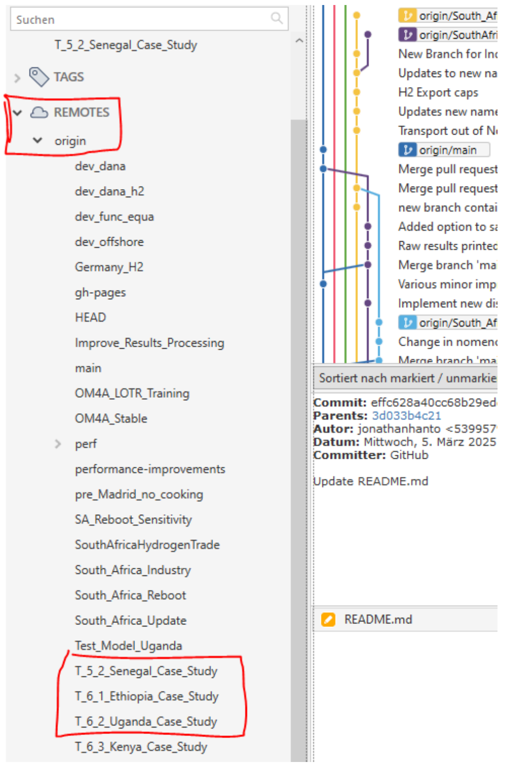
1.3 Switching Branches in SourceTree

To switch a branch in SourceTree, you simply open the sourcetree application on your machine. Press Fetch to get the newest updates. The **bold** branch on the left is the branch you are currently in (**pre_madrid_no_cooking** in the example below). You can change the branch by double-clicking another one.



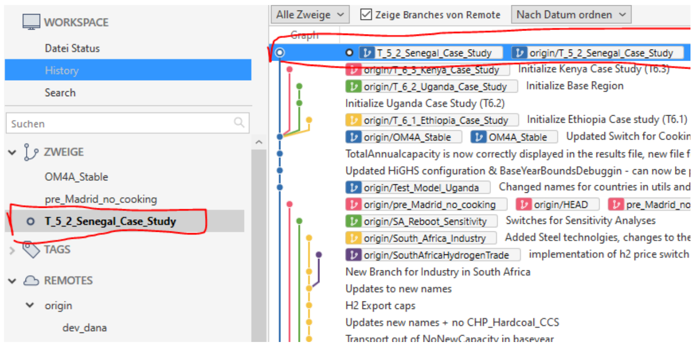
Updated HiGHS configuration & BaseYearBoundsDebuggin - can now be printed
origin/Test_Model_Uganda Changed names for countries in utils and changes equation
pre_Madrid_no_cooking origin/pre_Madrid_no_cooking origin/HEAD Up
origin/SA_Reboot_Sensitivity Switches for Sensitivity Analyses
origin/South_Africa_Industry Added Steel technolgies, changes to the Hourly Data to ad
origin/SouthAfricaHydrogenTrade implementation of h2 price switch which allows to er
New Branch for Industry in South Africa
Updates to new names
H2 Export caps
Updates new names + no CHP_Hardcoal_CCS
Transport out of NoNewCapacity in baseyear
origin/main Dev danar (#15)
Merge pull request #13 from GENeSYS-MOD/feat/dispatch
Merge pull request #2 from jonathanhanto/South_Africa_Update
new branch contains implementation of hydrogen trade through a new technology
Added option to save raw results in both CSV and TXT Dispatch results plotted in subfolder A
Raw results printed to subfolder and docstrings: * add docstrings to new structures * the resu

To find the branches prepared for the case-studies, you have to go to Remotes → origin (see image below) and scroll down until you find the name of the case-study and then double-click it.



Commit: effc628a40cc68b29ed
Parents: 3d033b4c21
Autor: jonathanhanto <539957
Datum: Mittwoch, 5. März 2025
Committer: GitHub

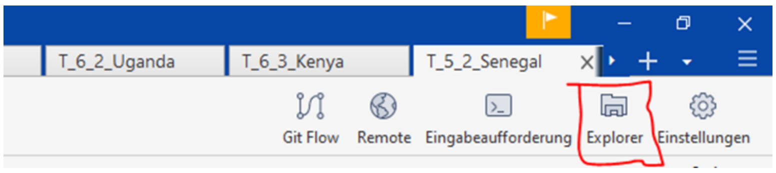
Update README.md



T_5_2_Senegal_Case_Study origin/T_5_2_Senegal_Case_Study
origin/T_6_3_Kenya_Case_Study Initialize Kenya Case Study (T6.3)
origin/T_6_2_Uganda_Case_Study Initialize Base Region
Initialize Uganda Case Study (T6.2)
origin/T_6_1_Ethiopia_Case_Study Initialize Ethiopia Case study (T6.1)
origin/OM4A_Stable OM4A_Stable Updated Switch for Cookin
TotalAnnualcapacity is now correctly displayed in the results file, new file f
Updated HiGHS configuration & BaseYearBoundsDebuggin - can now be
origin/Test_Model_Uganda Changed names for countries in utils and
origin/pre_Madrid_no_cooking origin/HEAD pre_Madrid_no
origin/SA_Reboot_Sensitivity Switches for Sensitivity Analyses
origin/South_Africa_Industry Added Steel technolgies, changes to the
origin/SouthAfricaHydrogenTrade implementation of h2 price switch
New Branch for Industry in South Africa
Updates to new names
H2 Export caps
Updates new names + no CHP_Hardcoal_CCS
Transport out of NoNewCapacity in baseyear

To ensure that all changes to the model are properly implemented, download the new run-file from Link Run File.

After you have switched branches, you will still need to create a new project, as some of the packages might interfere with the newest version and we have had multiple problems when simply updating all the packages. To do this, you will simply follow the installation Guide and do a new install of everything but use the new branch. If you cannot find the correct path, you can go into Sourcetree and click on “explorer”, which will open the folder that contains the mode.



Again, always make sure, you go into package mode "]" after starting GENeSYS-MOD and activate your environment using "activate ." in the Terminal.

1.4 Updating Subsets in Input Data file

The latest version of GENeSYS-MOD no longer requires a separate *Subsets* Excel file. Instead, the relevant data—consisting of two specific parameters—is now fully integrated into the main input data file.

To update the input data:

1. Open the older **`Tag_Subsets.xlsx`** file.
2. Locate and copy the data from the following parameters:
   - **`Par_TagTechnologyToSubsets`**
   - **`Par_TagFuelToSubsets`**
3. Open the main input data file.
4. Navigate to the same parameters (**`Par_TagTechnologyToSubsets`** and **`Par_TagFuelToSubsets`**) in this file.
5. Paste the copied data into these parameters, replacing any existing content.

This process ensures that the necessary subset tagging is correctly transferred to the updated input format.

2. How to run GENeSYS-MOD

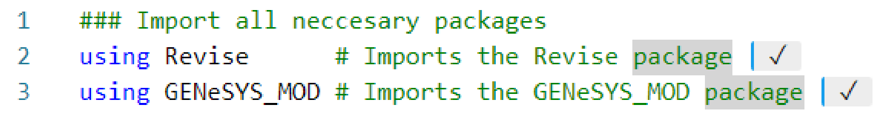
2.4 The run-file

You can find an exemplary run-file in the following location: Link Run File.
The run file consists of multiple sections. Section one is used to import all the packages needed, including the GENeSYS_MOD package. The second section defines the data directories and allows the user to input their data files. The third section is calling the genesysmod function, executing a model run based on the defined parameters. In the first two sections, each line can be executed individually. If a text is highlighted, it needs to be adapted by the user to match the name of the file. E.g., The input_file needs to be named exactly as the input file the user wants to use for the model run.

`using Revise`
`using Gurobi`
`using HiGHS`
`using Ipopt`
`using CSV`
`using DataFrames`
	Imports all needed packages.

`data_dir = "Inputdata"`
`result_dir = "Results"`
`input_file = "inputdata_UGANDA_old"`
`hourly_file = "Hourly_Data_UGANDA"`
	"Inputdata" and **"Results"** refer to folders within the main project directory.
	The files **input_file** and **hourly_file** should be correctly named as specified by the user.
	These files must be placed inside the **Inputdata** folder in order to be accessed by the model. Model results will be written into the folder specified in the results_dir.

Execute each line by pressing shift + enter. After a successful execution each line will show a mark. Make sure you are in the correct environment you have created in the beginning. You can enter package mode by typing "]" in the command line. This will usually enter the environment created with the folder.



```
### Import all neccesary packages
using Revise      # Imports the Revise package | ✓
using GENeSYS_MOD # Imports the GENeSYS_MOD package | ✓
```

Before running the by executing the genesys-mod function, some more model-run specific parameters need to be specified. In the following they are clustered by function they have:

`gmod = genesysmod(;`
`elmod_daystep = Int16(0),`
`elmod_hourstep = Int16(0),`
`elmod_nthhour = Int16(1444)`
	This section defines the model's hourly resolution. You must use either daystep with hourstep, or nthhour. The unused option(s) must be set to zero.
	
	Option 1: daystep and hourstep daystep defines how many full days (24h) to skip. hourstep sets the hour within the first selected day.

	The model starts at daystep * 24 + hourstep and adds daystep * 24 + hourstep each step. Example: daystep = 2, hourstep = 6 → hours: 54, 108... daystep = 1, hourstep = 0 → hours: 24, 48, 72... **Option 2**: nthhour Selects every nth hour out of the 8760 hours in the year. nthhour = 24 → hours: 0, 24, 48... (same hour each day) nthhour = 25 → hours: 0, 25, 50... (shifts through all 24h) **General advice:** To avoid bias, especially for time-sensitive technologies like solar PV, you should choose time steps that cycle through different hours of the day. Repeating the same hour (e.g., always midnight) may miss key effects and reduce model accuracy. Important: Use only one method. If nthhour > 0, both daystep and hourstep must be 0.

`gmod = genesysmod(;`
`elmod_daystep = Int16(0),`
`elmod_hourstep = Int16(0),`
`elmod_nthhour = Int16(1444),`
	This section is to define the hourly resolution the model uses. You either have to use daystep+hourstep or nthhour. The other one(s) have to be set to zero. In general, it is sensible to use a step is

`year = Int16(2018),`	Define the baseyear as used in the data

`solver = HiGHS.Optimizer,`
`DNLPsolver = Ipopt.Optimizer,`
`threads = Int16(8),`
	This section is to define the solver. Additionally, the threads used can be defined

`inputdir = data_dir,`
`data_file = input_file,`
`hourly_data_file = hourly_file,`
`resultdir = result_dir,`
	This section ensures the directories are correct. They are defined before the genesysmod function (see above)

`emissionPathway = "Name",`
`emissionScenario = "Name",`
	Here the user can choose names for the emissionspathway and the emissionscenario, for each model run the names can be changed to ensure the user knows the runs details.
	Important: if the name is kept the same, the previous results will be overwritten

`model_region = "UG",`	The model_region can be freely named. The data_base_region,

`data_base_region = "UG-C",`	however, needs to be named the same as the base region used in the input data files!!
`switch_base_year_bounds = Int16(1),`	This switch allows the user to turn on (1) or off (0) the base_year_bounds. IT should be on to ensure the base year is calibrated correctly. For debugging you can use the switch_base_year_bounds_debugging tool that will be explained below and in more detail in section 2.
`switch_processed_results = Int8(1),`	This switch allows the user to turn on (1) or off (0) the use of processed results, this will create pre-processed results files for easier analysis. Otherwise, the model will output raw result data.
`switch_ccs = Int16(0),`	This switch allows the user to turn on (1) or off (0) the use of ccs technologies in the model.
`switch_investLimit = Int16(1),`	This switch allows the user to use (1) or turn off (0) investlimit, which ensures that both investment and output evolve smoothly.
`switch_cooking=Int8(1)`	This switch allows the user to use (1) or turn off (0) Cooking as defined in the previous workshops. The switch mainly enables the read in of data specific to cooking. If the user has no cooking data, the switch needs to be put to “0”.
`switch_base_year_bounds_debugging=Int8(0)`	This switch allows the user to use (1) or turn off (0) the base_year_bounds_debugging function explained in more detail in section 2.
`switch_iis=Int8(0)`	This switch allows the user to use (1) or turn off (0) the creation of an iis if the model goes infeasible. It will be explained in more detail in section 3.
`Switch_infeasibility_tech = NoInfeasibilityTechs()`	Allows the user to enable (WithInfeasibilityTechs()) or to disable (NoInfeasibilityTechs()) the usage of the infeasibility techs to

	help figure out why the model becomes infeasible. This will be explained in more detail in section 3.

3. Calibrating the Model for the Baseyear & 2025

Once you have all the data ready, you still need to calibrate it so the model runs under its various equations and constraints, especially in the base year, where outputs must closely match historical values. This iterative process involves toggling specific switches and settings to identify which elements cause errors or infeasibilities and then adjusting the data or configuration accordingly. Although there's no fixed sequence of steps, the following sections outline key concepts and examples to guide you through effective calibration.

3.1 **No New Capacity Constraint**

In the model setup, **no new capacities are allowed to be built in the base year (e.g., 2018)**. This is enforced programmatically in genesysmod_bounds.jl to ensure the system configuration in the starting year reflects only pre-existing infrastructure and avoids unrealistic deployment of new technologies during the calibration phase. In this step, we have to calibrate the power sector and then create the ResidualCapacities for heating and Transport. To do so, we disable the capacity constraints, let the model build the capacity (also see section Par_ResidualCapacity in Inputdata Guide), put this info into Par_RedisualCapacity and then enable the constraints.

For this step, you want to have

- RegionalBaseYearproduction for Heating + Power
- ModalSplit Data
- Baseyearboundsdebuggin to zero
- Baseyearbounds to 0
- switch_infeasibility_tech = NoInfeasibilityTechs()
- The code for Genesysmod_bounds.jl open

3.1.1 Disabling the No New Capacity Constraint for the BaseYear

In line 179, the user can find the following code:

for t ∈ intersect(Sets.Technology,
vcat(Params.Tags.TagTechnologyToSubsets["**Transformation**"],
Params.Tags.TagTechnologyToSubsets["**PowerSupply**"],
Params.Tags.TagTechnologyToSubsets["**SectorCoupling**"],
Params.Tags.TagTechnologyToSubsets["**StorageDummies**"],
Params.Tags.TagTechnologyToSubsets["**Transport**"],
Params.Tags.TagTechnologyToSubsets["**CHP**"]))

By deleting a line (eg. “Params.Tags.TagTechnologyToSubsets[“Transformation”]”) the user can disable the constraint for all technologies in the Subset Transformation. To see all technologies that are part of this subset, you can check the parameter TagTechnologyToSubsets in your input data file.

3.1.2 Calibrating the Power Sector in the BaseYear

First, we want to ensure that the power sector is calibrated. By removing the Transformation & Transport lines and saving the file, you ensure that missing residual capacity won’t block the run at this point, allowing you to calibrate the Power sector:

Params.Tags.TagTechnologyToSubsets["**PowerSupply**"],
Params.Tags.TagTechnologyToSubsets["**SectorCoupling**"],
Params.Tags.TagTechnologyToSubsets["**StorageDummies**"],
Params.Tags.TagTechnologyToSubsets["**CHP**"]))

If the model now runs with PowerSupply enabled, proceed to section 3.1.3.

If the model goes infeasible, this means, the model is not able to satisfy power demand with what was given by the user (the main reason usually is residual capacity but also missing TradeCapacity could be an issue)
You now have two options to pinpoint the issue:

1. Turn on infeasibilitytech in the run-file to find what is causing the issue. After enabling infeasibilitytech by setting switch_infeasibility_tech = WithInfeasibilityTechs(), the model should run without going infeasible. If the model goes infeasible despite having infeasibilitytech on, go to section 4 to see what else might cause the issue. After finishing the model run, you can access the output_capacity & output_annual_production files to look for the infeasibility_tech that shows you the region, fuel, and production amount in the output_annual_production and the new capacity it builds in the output_capacity file. This helps to understand where there is missing capacity. The user still needs to figure out where capacity is missing and might need to revisit some data assumptions.
2. If the infeasibility_tech is somewhat inconclusive, you can also disable PowerSupply in the Baseyearbounds, leaving you with:
   Params.Tags.TagTechnologyToSubsets["**SectorCoupling**"],
   Params.Tags.TagTechnologyToSubsets["**StorageDummies**"],
   Params.Tags.TagTechnologyToSubsets["**CHP**"])).
   After running the model, you can access the ouput_capacity results file to check the new capacity built in 2018 in the Power sector. This also shows where capacity was needed. As mentioned before, this is an iterative process and asks for the user to recheck the input data and ensure trade etc. is properly enabled.

3.1.3 Base Year Bounds Debugging Power Sector

You can now run the model with baseyearbounds=1, if the model runs without any infeasibilities, you can forward to section 3.1.4. If the model goes infeasible, follow section 3.2 to debug the base year and ensure power production is met as outlined in the Parameter RegionalBaseYearProduction

3.1.4 Manual Residual Capacity for Heating and Transport

The subset Transformation contains all the heating technologies and the subset Transport contains all transport technologies. By deleting these lines, saving the file and running the model, we can let the model create capacities in the base year, take the results and put them into the Residual Capacities, then put the lines back in the code, save and run the model. See Section Par_ResidualCapacity in the Input Data Guide for further information. It is important that baseyearbounds = 1 as we want the capacities to be built to fit what is expected to be produced in the base year. Also, to ensure the model can run, it can be sensible to increase the capacities by a few percent (eg. 5%) to ensure the model has sufficient capacity to run under different hourly set-ups.

Should the model build capacity that is not being used (You can compare by checking output_capacity & output_annual_production) you do not need to input that as residual capacity.

3.2 Base Year Bounds Debugging

In the base year, the model aims to reproduce historical values **without allowing new capacity construction**. It assumes existing (residual) capacity is sufficient to meet historical demand. Therefore, the model must be carefully calibrated so that production aligns with both capacity and observed data.

!BaseYearBoundsDebuggin should be done after the model is calibrated so that all No New capacity Constraints are enabled and the model is feasible!

When *`switch_base_year_bounds = 1`*, the model **strictly enforces** constraints tying production to historical values. However, during initial calibration, this can lead to infeasibilities if capacity and production data are misaligned.

To help with this, *`switch_base_year_bounds_debugging`* can be enabled. It relaxes the bounds by allowing slack variables to take non-zero values, helping identify **where and by how much** the model deviates from historical data. This highlights mismatches between demand and capacity and helps with data calibration.

Slack variable values (`BaseYearBounds_TooHigh` and `BaseYearBounds_TooLow`) indicate whether production is falling short or exceeding what's expected. These diagnostics help sync capacity and demand data and improve model accuracy.

When `Switch.switch_base_year_bounds == 1`, the model applies upper and lower bounds on `ProductionByTechnologyAnnual`, based on `RegionalBaseYearProduction`, ensuring results are close to historical values.

Code implementation can be found in `genesysmod_equ.jl`.

3.2.1 What does `switch_base_year_bounds_debugging` do?

This switch activates debugging mode for base year bounds:

```
if Switch.switch_base_year_bounds_debugging == 0

    JuMP.fix(Vars.BaseYearBounds_TooHigh[r,t,f,y], 0; force=true)

    JuMP.fix(Vars.BaseYearBounds_TooLow[r,t,f,y], 0; force=true)

end
```

- When `debugging == 0`: slack variables are fixed to 0 — **strict enforcement**.
- When `debugging != 0`: slacks are free — allowing the model to show **where deviations occur**.

This is useful to test model feasibility or identify structural data issues.

3.2.2 What do `BaseYearBounds_TooLow` and `TooHigh` do?

These are **positive slack variables** used to relax production bounds. Results are available in the output files when `Switch.switch_base_year_bounds == 1`.

Lower Bound (BYB1):

```
Production >= BaseYearProduction * (1 - Slack%) - TooHigh
```

- Ensures production isn't too low.
- `TooHigh` shows how far below the target the model drops, if needed.

Upper Bound (BYB2):

```
Production <= BaseYearProduction + TooLow
```

- Ensures production doesn't exceed expectations.
- `TooLow` allows overshooting when necessary.

Summary:

- `TooHigh` → how much below target the model goes.
- `TooLow` → how much above it goes.

With debugging **off**, both slacks are fixed to zero, making the bounds strict.

For debuggin, BYB1 is key as the model errors when production cannot meet demand in the baseyear. BYB2 usually indicates that there is not enough production in general to meet SpecifiedAnnualDemand.

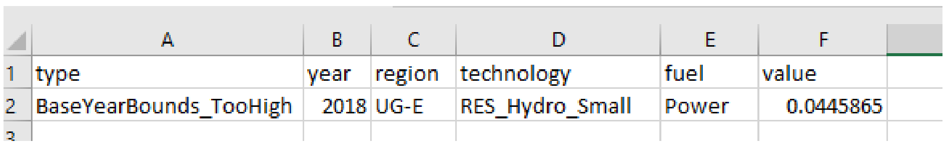
Example:

When running the model with `baseyearbounds=0`, it completes successfully. However, enabling `baseyearbounds=1` may lead to infeasibility due to inconsistencies in base year capacity data.

To identify and address the root cause:

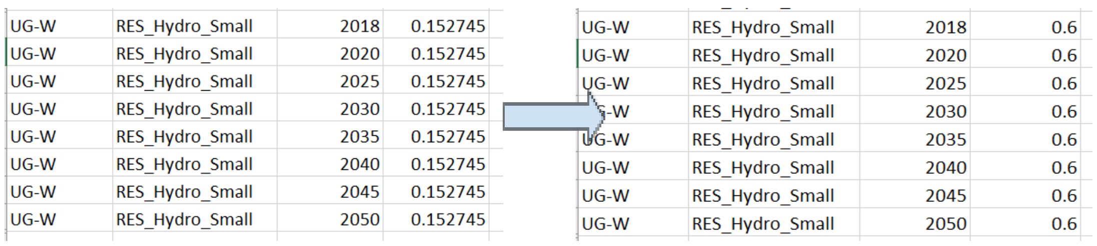
1. Activate Debugging Mode
   Set `switch_base_year_bounds_debugging=1`. This allows the model to bypass strict base year constraints temporarily and helps pinpoint problematic technologies or regions without causing infeasibility.

2. Identify Problematic Entries
   Run the model and examine the output file "output_BYB_bounds.csv" in the Results folder. Compared to previous versions, the value indicated in the results now shows how much production is missing for TooLow & TooHigh. This can be used to calculate how much more residual capacity is needed.



	A	B	C	D	E	F
1	type	year	region	technology	fuel	value
2	BaseYearBounds_TooHigh	2018	UG-E	RES_Hydro_Small	Power	0.0445865

3. Adjust Residual Capacity
   Modify the input data to increase the `residual_capacity` for the affected technology and region. Re-run the model to confirm whether the issue has been resolved.

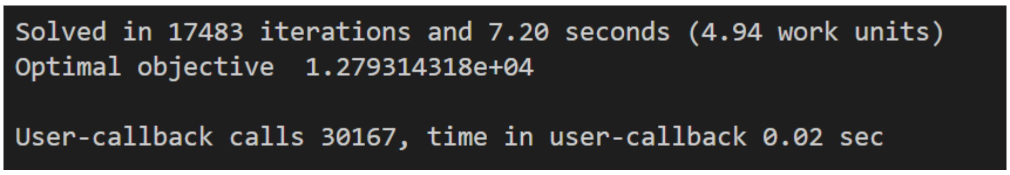


UG-W	RES_Hydro_Small	2018	0.152745
UG-W	RES_Hydro_Small	2020	0.152745
UG-W	RES_Hydro_Small	2025	0.152745
UG-W	RES_Hydro_Small	2030	0.152745
UG-W	RES_Hydro_Small	2035	0.152745
UG-W	RES_Hydro_Small	2040	0.152745
UG-W	RES_Hydro_Small	2045	0.152745
UG-W	RES_Hydro_Small	2050	0.152745

UG-W	RES_Hydro_Small	2018	0.6
UG-W	RES_Hydro_Small	2020	0.6
UG-W	RES_Hydro_Small	2025	0.6
UG-W	RES_Hydro_Small	2030	0.6
UG-W	RES_Hydro_Small	2035	0.6
UG-W	RES_Hydro_Small	2040	0.6
UG-W	RES_Hydro_Small	2045	0.6
UG-W	RES_Hydro_Small	2050	0.6

4. Repeat if Necessary
   Continue reviewing "output_BYB_bounds.csv" until all problematic entries are addressed.

5. Finalize the Fix
   Once all entries are cleared, set `switch_base_year_bounds_debugging=0` and run the model again with `baseyearbounds=1`. It should now complete successfully.



```
Solved in 17483 iterations and 7.20 seconds (4.94 work units)
Optimal objective  1.279314318e+04

User-callback calls 30167, time in user-callback 0.02 sec
```

Note: The capacities in the example may have been increased substantially to expedite troubleshooting. For more accurate modeling, consider gradually adjusting the values to identify the minimum capacity that ensures feasibility, staying closer to the original input data.

3.3 Calibrating 2025 & other Years

Given that 2025 is not the base year, but it is already the current year, the user might want to calibrate this year as well. Although it cannot be used as a base year since data is not yet fully available, some trajectories might need to be fixed. This is because the model might take a vastly different route when optimizing between 2018 and 2025 compared to what actually happened. The user can do this in the Input Datafile using mainly four parameters:

Par_TotalAnnualMaxActivity - To set an upper bound for production from one Technology
Par_TotalAnnualMinActivity - To set a lower bound for production from one Technology
Par_TotalAnnualMaxCapacity - To set an upper bound for Capacity expansion
Par_TotalAnnualMinCapacity - To set a lower bound for capacity expansion

4. Dealing with errors

This section aims to provide users with solutions for dealing with errors and infeasibilities. There is no single way to resolve a model infeasibility—multiple factors may be at play, and you must pinpoint the root cause. Turning off the “No New Capacity” constraint for specific subsets can help address capacity‑mismatch issues. Likewise, toggling individual switches on or off can reveal whether a particular feature is driving the infeasibility, thereby isolating the problem. A few such options are explained in the following:

4.1 infeasibility-tech

When the infeasibility_tech switch is enabled, the model uses a “dummy” technology hooked up to every demand type with unlimited capacity but prohibitively high costs. That way, it can always fill any gap—but only if no real option exists—so whenever the solver uses it, the user can spot which demand or constraint is causing the infeasibility. It shows up as infeasibility_tech in the results file.

4.2 investLimit

When `switch_investLimit` is turned on, the model adds a suite of inter‑year constraints to ensure that both investment and output evolve smoothly and stay within the user’s specified growth limits. For each year after the base year:

- **Total capital investment** is capped at a fixed fraction of the remaining investment horizon (using the `InvestmentLimit` and a yearly scaling multiplier), so you can’t front-load or back-load spending.

- **New renewables capacity** in each region is limited by a scaled “maximum allowable” threshold (*`NewRESCapacity`* × regional capacity ceiling), preventing runaway build-outs.

- **Technology ramp-up and phase-out** are smoothed year-on-year: “phase-in” technologies must grow by at least a minimum rate, and “phase-out” technologies may only shrink by a maximum rate, both adjusted for changes in annual demand.

- **Overall fuel-based production growth** across all technologies is bounded by the *`ProductionGrowthLimit`*, so aggregate output can’t spike beyond a set percentage of last year’s level.

- **Storage-dummy technologies** receive their own growth cap (*`StorageLimitOffset`* added to the general production limit) to control how quickly energy storage can expand.

Together, these constraints enforce gradual, realistic investment and production paths in line with the model’s InvestmentLimit and production-growth settings.

If disabling the investment-limit constraint restores feasibility, it usually signals a demand–supply imbalance—your specified load or fuel requirements exceed the sum of available generation or supply technologies. It may also mean you’re bumping up against the *`ProductionGrowthLimit`* (5 % by default), which you can adjust in *`genesysmod_settings.jl`*. Finally, double-check that your `SpecifiedAnnualDemand` growth through 2050 isn’t unrealistically steep.

4.3 IIS

Turning on *`switch_iis`* will generate an IIS file if the model is infeasible (not supported by HiGHS). The IIS report can be confusing when the source of infeasibility is unclear, but it provides a useful guide to pinpoint the constraint or variable causing the issue. More info on Gurobi’s site;

“The 'compute IIS' results are designed to be examined and analyzed by human users, providing a starting point for understanding the problematic constraints in the model. The smaller the IIS, the more effectively it can pinpoint the specific issues in the model.”

5. Visualization

→ See the presentation from Madrid for more information on results Visualization

5.1 Get Tableau licences

Please follow the information below to update your activation key for Tableau or to download and use Tableau for the first time. You have to be connected to an academic institution to able to use Tableau.

Thank you for your interest in the Tableau for Teaching program. Below is a website (landing page link) for your upcoming class. Each student should go to the landing page to download Tableau and activate with the product key noted below. This key will allow your entire class to activate both Tableau Desktop and Tableau Prep Builder for the duration of the course. **The below key is valid for 30 activations and will expire on 7/31/2025.**

Please forward these instructions to students enrolled in Basic methods for energy modeling:

1. Sign into an existing Tableau.com account, or create a new account using your school-issued email
2. Once signed in, visit the Academic Quick Start page to download the latest versions of Tableau Desktop and Tableau Prep Builder
3. Activate with product key: **TCOX-03E9-3560-891E-33B1**
4. Already have a copy of Tableau Desktop installed? Update the license key in the application: Help menu → Manage Product Keys

Having trouble? The Academic Quick Start page includes FAQs and help articles.

Some helpful resources:

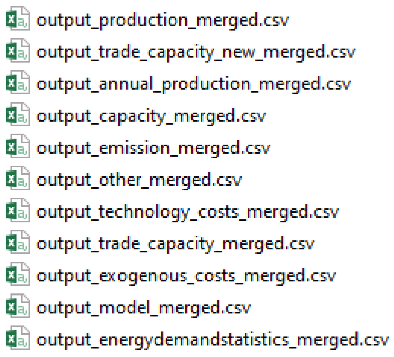
- Check out Tableau's Free Training Videos and Tableau Public's How-To Videos
- Join the Tableau Community and explore the Forums to search and ask questions
- For product support, review the Knowledge Base and Tableau Help platform

If you use a Mac or Linux operating system, you may need to install specific drivers to run Tableau Desktop and Tableau Prep Builder. You can find relevant drivers here. As a note, Tableau uses the Simba Spark driver to connect to Kyvos. These drivers are to be used for your instance of Tableau only and will be linked to your product key.

For further questions, you may review our FAQ page.

3.2 Merging Result Files using merge.bat

To enable multiple runs or scenarios to be combined into a single file for use in Tableau, a `merge.bat` script has been prepared. This script processes the different scenario result files and creates a new `.csv` file containing the results from each scenario.



output_production_merged.csv
output_trade_capacity_new_merged.csv
output_annual_production_merged.csv
output_capacity_merged.csv
output_emission_merged.csv
output_other_merged.csv
output_technology_costs_merged.csv
output_trade_capacity_merged.csv
output_exogenous_costs_merged.csv
output_model_merged.csv
output_energydemandstatistics_merged.csv

You can find the `merge.bat` file under **Merge_File**. To use it, simply place the script in the same folder as all the result files you want to merge and double-click the file. This will start the merging process

Once completed, new files with the prefix **"merged"** will appear in the folder. It is recommended to delete any previously merged files **before starting a new merge** to avoid duplication of data.

5.3 How to use Tableau

Please see Presentation_Workshop_Madrid_2024 for a presentation from the Workshop in Madrid to explain how to use Tableau. There are a lot of free online resources explaining the functionalities of the tool, simply search on google or YouTube.