Royal Institute of Technology

Division of
Energy Systems Analysis

Working Paper Series

DESA/15/10

OSeMOSYS Manual – Working with text files

Version of October 2015

Nandi Moksnes[*1], Manuel Welsch*[*1], Francesco Gardumi[2], Abhishek Shivakumar[1], Oliver Broad[1]*

Please send all comments and feedback you may have regarding this manual to General OseMOsys e-mail address as mentioned on osemosys.org. This will help us update and improve it.

[1] KTH Royal Institute of Technology, Stockholm, Sweden.
[2] Polytechnic University of Milan, Milan, Italy.

* Corresponding Authors
E-mail:nandi@kth.se,
manuel.welsch@gmx.at

KTH Royal Institute of Technology
Stockholm, Sweden

Table of content

Index for parameters

PURPOSE

The purpose of this paper is to help new OSeMOSYS users to understand how to set up an OSeMOSYS model from scratch, defining the model parameters in text files. However, this manual is also useful for users applying one of the OSeMOSYS interfaces that are currently being developed. This is because of its description of how to set up and debug an OSeMOSYS model and the definition of the various input parameters used in OSeMOSYS. For illustrative purposes, the demonstrations in this paper refer to a consistent sample dataset called "UTOPIA". The version referred to in this manual is OSeMOSYS_2013_05_10.

The paper describes how to get started with OSeMOSYS, using LEAP for modelling with OSeMOSYS, mapping the reference system, choice of units and data collection and parameter descriptions with examples from UTOPIA. There is also some tips for debugging the model if there is no feasible solution to the model.

Quite some of the descriptions have been compiled from the following two publications, which are therefore not referenced throughout this document whenever language is taken from them:

- *"OSeMOSYS: The Open Source Energy Modeling System, An introduction to its ethos, structure and development" by M. Howells et al. in 2011*
- *"Modelling elements of Smart Grids – Enhancing the OSeMOSYS (Open Source Energy Modelling System) code" by* Welsch, M. et al. in 2012

INTRODUCTION TO OSEMOSYS

At present there exists a useful, but limited set of accessible energy system models. These tools often require significant investment in terms of human resources, training and software purchases in order to apply or further develop them. Unlike long established energy systems (partial equilibrium) models[1] OSeMOSYS potentially requires a less significant learning curve and time commitment to build and operate. Additionally, by not using proprietary software or commercial programming languages and solvers, OSeMOSYS requires no upfront financial investment.

OSeMOSYS is designed as a tool to inform the development of local, national and multi-regional energy strategies. It covers all or individual energy sectors, including heat, electricity and transport. It is a deterministic linear optimisation model and minimises the total discounted costs. Mixed integer programming may be applied for certain functions, like the optimisation of discrete power plant capacity expansions.

The model is driven by exogenously defined demands for energy services. These can be met through a range of technologies which draw on a set of resources, defined by their potentials and costs. On top of this, policy scenarios may impose certain technical constraints, economic realities or environmental targets. As in most long-term optimisation models, OSeMOSYS in its standard configuration assumes perfect foresight and perfect competition on energy markets.

The model is characterised by a wide and flexible technology definition. A technology comprises any fuel use and conversion, from resource extraction and processing to generation, transmission and distribution, and appliances. It could therefore refer to, for example, an oil refinery, a hydropower plant or a heating system. A technology can be defined to consume and produce any combination of fuels. Each technology is characterised by numerous economic, technical and environmental parameters, for example, investment and operating costs, efficiencies, availabilities, and emission profiles.

[1] Models such as MARKAL/TIMES [8], MESSAGE [9], PRIMES [10], EFOM [6] and POLES [7].

The OSeMOSYS model code is written in GNU MathProg, a high level mathematical programming language. The open source solver GLPK may be used for the mathematical optimisation of the model. Both the OSeMOSYS model and the GLPK solver do not require any upfront financial expenditure. GLPK can as well produce an MPS file for use with a more powerful solver[2]. Diverging from commonly applied programming conventions, rather long parameter and variable names are used in OSeMOSYS. This ensures that formulas can be read in a rather self-explanatory manner and simplifies the familiarisation with the OSeMOSYS code for those new to this modelling tool.

In its extended version, OSeMOSYS comprises just above 400 lines code. In 2013, a parallel shortened version of OSeMOSYS have been released. The merging of equations significantly improved the performance without affecting the model's data requirements or results at the price of a reduced readability of the code.

OSeMOSYS is developed in collaboration with a range of institutions, including the International Atomic Energy Agency (IAEA), the United Nations Industrial Development Organisation (UNIDO), KTH Royal Institute of Technology, Stanford University, University College London (UCL), University of Cape Town (UCT), Paul Scherrer Institute (PSI), Stockholm Environment Institute (SEI), and North Carolina State University.

Several user interfaces are currently available or under development. For example, OSeMOSYS is well integrated into LEAP, which applies a limited set of OSeMOSYS' optimisation features for power plant capacity expansion planning. Two alternative interfaces are currently being developed by KTH. One is a simplified Excel-based interface for teaching purposes, while the other will fully support the complete functionality of OSeMOSYS. However, at this stage, if the full scope of OSeMOSYS is required or if adjustments are necessary, OSeMOSYS is best set up as text files. This manual provides an introduction on how to do so.

While the development of OSeMOSYS initiated in 2008, its first detailed description was published by Howells et al. in 2011 and a subset of the expansions described in Welsch, M., et al. in 2012 led to its current version. This version is available for download at www.osemosys.org.

[redacted] GETTING STARTED WITH OSEMOSYS

OSeMOSYS models comprise a model file and a data file. This manual explains how to set up the data text file. The Command Prompt is then used to call the solver to optimise the model described in the model file, given the data provided in the data file. This produces a results text file and a Comma Separated Values (csv) file, which can easily be opened in Excel and presents the main outputs of the model.

3.1 A FIRST MODEL RUN

To do a first model run, start by downloading the OSeMOSYS files from osemosys.org which consists of two OSeMOSYS files, one short and one long version (both of which produce the same results, see next section), and one example file called UTOPIA.

The main steps to perform a model run are outlined in the following:

1) **Getting started with the UTOPIA example in OSeMOSYS**
    a. Download GLPK. GLPK is free and open-source software http://sourceforge.net/projects/gnuwin32/files/glpk/4.34/glpk-4.34-setup.exe/download.
    b. Once you have downloaded the setup.exe, execute the file. All this does is to extract files in a folder of your choice (referred to as *** in the following).
2) **The OSeMOSYS model file and the UTOPIA data file**

[2] For example, GUROBI offers free academic licenses.

Cut and paste the OSeMOSYS_201*_*.txt and UTOPIA.txt file into the ***\GnuWin32\bin directory. (Remember that you set which directory this is after executing the glpk-*.**-setup.exe file.)

3) **Open the command prompt**

The command prompt can be opened by clicking in Windows on Start, Run…, and then typing 'cmd'. (There are other ways of doing this but the latter is one of the easiest.) The command prompt will have some text to indicate the current directory to which you can send 'prompts'. If you are not in the same directory as the one where GLPK stored, then you need to change the directory that contains 'glpsol.exe' (***\GnuWin32\bin). If you extracted it to (the default) 'program files', you need to go to C:\Program Files\GnuWin32\bin. Open the command prompt and 1) return to the root by typing "cd/" then 2) navigate to the folder that the files are stored in by typing "cd C:***\GnuWin32\bin then hit the return key (if you need help on command prompt, type 'help' and possible command options will be listed).

4) **Running the UTOPIA example in OSeMOSYS using 'glpsol'**

Now that you are in the ***\GnuWin32\bin directory in the command prompt, and you have OSeMOSYS_201*_**_*.txt and UTOPIA.txt pasted in the same directory you can run the UTOPIA example in OSeMOSYS. To do this type: "**glpsol -m OSeMOSYS_201*_**_**.txt -d UTOPIA_201*_**_**.dat -o results.txt**" **All characters are important including space.** This invokes a command (glpsol) to take the model file (OSeMOSYS_201*_**_*.txt) and associated data input file (UTOPIA.txt) to produce an output file with a full set of results (result.txt). The results file is particularly large – even for a simple problem. To make life easier, therefore OSeMOSYS is programmed to produce a summary results file called SelectedResults.csv. This, as well as the full results file will appear in the ***\GnuWin32\bin directory after a model run.

A list of other possible command options can be found in the command prompt by typing glpsol –help.

5) **Errors**

Glpsol will display an error message if it does not understand the input files, and it will also tell you in which line number there is a conflict.

6) **Output**

When glpsol runs successfully, it prints a status line. Each line will look similar to the following: * 4: objval = 1. 563750000e+002 infeas = 0.000000000e+000 (0). '*' means that a basic feasible solution has been found, '4:' means that there have been 4 iterations to find a solution so far. 'objval' shows the current objective value, and 'infeas' shows the amount of infeasibility. When a feasible solution has been found, its value will be either 0 or a very small number. For more information on this please read this the documentation on GNU Linear Programming from Rodrigo Ceron Ferreira de Castro here: http://www.osemosys.org/uploads/1/8/5/0/18504136/ceron_-_2006_-_the_gnu_linear_programming_kit,_part_1_-_introduction_to_linear_optimization.pdf.

7) **Solution**

To see the full solution, use a text editor to open results.txt. (For example, Notepad or Notepad++, see Section 3.6 on Supportive Programmer and Documentation). Recall that the solution file will be found in the directory ***\GnuWin32\bin. The summary solution file for OSeMOSYS is a comma separated file called SelectedResults.csv. A csv file is conveniently opened in a spreadsheet and by using the 'comma' option when asked how data is 'delineated'. The selected results file produces tables of the following outputs. (The units indicated are specific to the UTOPIA example. Other units may be defined by the user when setting up a new data file):

- Total emissions, by type and region (emissions units, Mton)
- Total costs, by region (currency units, m$)
- The (time independent) demand for each: energy carrier (this is zero if no demand was entered), region and year (energy units, PJ)
- The (time dependent) demand for each: energy carrier (this is zero if no demand was entered), time slice, region and year (energy units, PJ)

- The (time dependent) production for each energy carrier timeslice, region and year (energy units, PJ)
- The total annual capacity of each technology by region (capacity units, GW)
- The new investment in capacity for each technology for each year by region (capacity units, GW)
- The annual production by each technology of each energy source by region (energy units, GW)
- The annual use by each technology of each energy source by region (energy units, PJ)
- Annual emissions, by species and region (emissions units, Mton)
- Annual emissions by technology, species and region (emissions units, Mton) [1].

If you have problems with running the files from the ***\GnuWin32\bin directory (due to administration rights), redirect the .txt you open and results file in a different directory. Remember that you then need to write the complete path (e.g., C:\Users\user001\Documents\OSeMOSYS_201*_**_*.txt) in the Command Prompt or change to the new folder before running the model.

3.2 USE OF THE FULL OR THE SHORT VERSION OF OSEMOSYS

When downloading OSeMOSYS, a full version and a short version are available. The long version is easier to read and understand, as it includes more simple, user-friendly equations. The shorter version was developed by merging equations from the full version. This eliminates the need to calculate and store intermediate values and results in faster calculation times. While it produces the same key outputs, it is harder to understand the individual equations. The full version computes more variables, which help the user in understanding what the model actually does. The full version is therefore as well particularly helpful to test modifications and enhancements to the main code. Therefore, the use of the full version is recommended unless the calculation times get prohibitively long.

Problems which cannot be solved with the full code due to time or RAM constraints can be solved with the short code. It is easy to check if the RAM is a limiting constraint: during the computation, it is sufficient to open the system task manager (Ctrl + Alt + Delete) and check if the RAM use is close to saturation. The RAM and time constraint may become binding when the model gets too complex. This may happen, e.g., when a multi-regional model with many technologies and time slices is set up. In such cases, the shorter version is recommended.

The reduction in the number of equations in the shorter version translates into the generation of a smaller matrix to be solved. This significantly reduces the memory usage (~10x) and the processing time (~5x). The short version of the OSeMOSYS code contains only the essential equations required for running the model. However, all the previous equations have been left as before, and "commented out" to better understand the methodology followed to shorten the code. It is important to note that the shortening of the code does not change any aspect of the functionality of OSeMOSYS. Further, there are no special formatting requirements for the data file required to run the short version instead of the full version. Both the regular and the short versions of OSeMOSYS are developed and released as parallel versions simultaneously.

3.3 MAPPING THE REFERENCE ENERGY SYSTEM

When developing a model in an optimisation tool like OSeMOSYS, the energy system needs to be mapped to identify all the relevant technologies that will be involved. To the left in the model the technologies categorised as primary energy resources are mapped (shown in boxes in Figure 1). The lines represent energy carriers, e.g., crude oil, coal etc. Moving from the left to right the energy carriers are transformed through different technologies to ultimately meed a final demand for energy (services), presented by the technologies at the very

right hand side of Figure 1. The Reference Energy System (RES) can be seen as a table of content for the energy model. Figure 1 illustrates the RES for a demonstration case referred to as UTOPIA. The chosen acronyms are freely chosen by the user and can therefore differ from one model to the next. In the UTOPIA RES for example, E01 is a coal fired power plant and RHE is Residential Heating Electricity [2] [3].

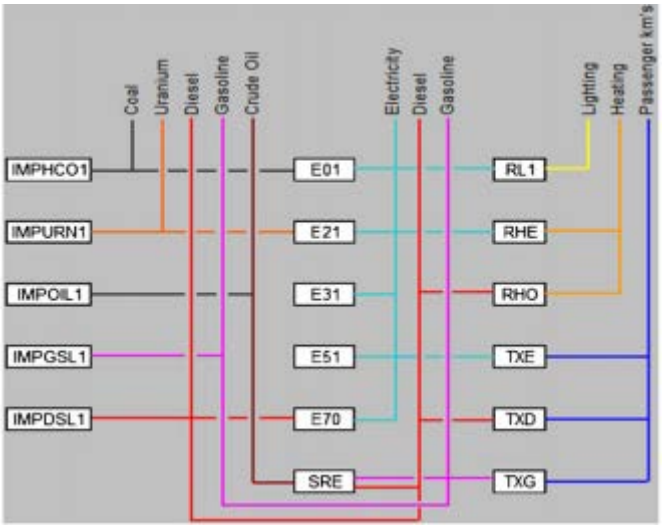

Figure 1 The reference system in UTOPIA [2].

KEY TO THE REFERENCES SYSTEM IN UTOPIA

E01 is a power plant which consumes coal (IMPHCOL).
E21 is a nuclear power plant that consumes uranium (IMPURN1).
E31 is a hydro power plant.
E51 consumes electricity and generates electricity.
E70 is a power plant that consumes diesel (IMPDSL1).
RHE is residential electricity heating that consumes electricity.
RHO is residential heating that consumes diesel (IMPLDSL1).
RHL is residential lighting that consumed electricity.
TXG is transport in passenger km consuming gasoline (IMPGSL1).
TXD is transport in passenger km consuming diesel (IMPDSL1).
TXE is transport in passenger km consuming Electricity [2].

3.4 CREATING AN INPUT FILE

To create the input data for an optimisation run you can set up the model directly in a text editor like Notepad++. It is advisable to start with a small model and build it up step wise. This will simplify the debugging (see Section 5 Debugging the model). It is further advisable to back up working versions of model data files by saving them in a folder of your choice. The Utopia input file provided with the downloaded model code might serve as a useful starting point to see how data needs to be correctly formatted. Alternatively, LEAP has also proven useful to write an OSeMOSYS data file.

HOW TO USE **LEAP** TO OPTIMISE THE ELECTRICITY SYSTEM

LEAP, the Long-range Energy Alternatives Planning System, is a widely used medium- to long-term modelling tool for integrated resource planning. It is applied to assess climate change mitigation strategies and analyse energy policies by investigating yearly capacity expansions. The underlying dispatch of technologies is calculated for each user-define time slice within a year. LEAP is a simulation tool, but uses OSeMOSYS for the

optimisation of power system investments. It is well documented, free for developing countries, and online support is provided by Stockholm Environment Institute (www.energycommunity.org).

In LEAP it is possible to optimise the LEAP branch Transformation\Electric Generation using OSeMOSYS. To do this follow the **exercise 6.5** in the LEAP Training Exercises found here: http://www.energycommunity.org/documents/LEAPTrainingExerciseEnglish2014.pdf.

HOW TO USE LEAP TO WRITE A DATA FILE, WHICH CAN THEN BE MODIFIED IN NOTEPAD++

When developing the model according to exercise 6.5 and run the optimisation in LEAP, the data file that is used for the OSeMOSYS optimisation within LEAP will be saved in your LEAP folder under the name OpData**:

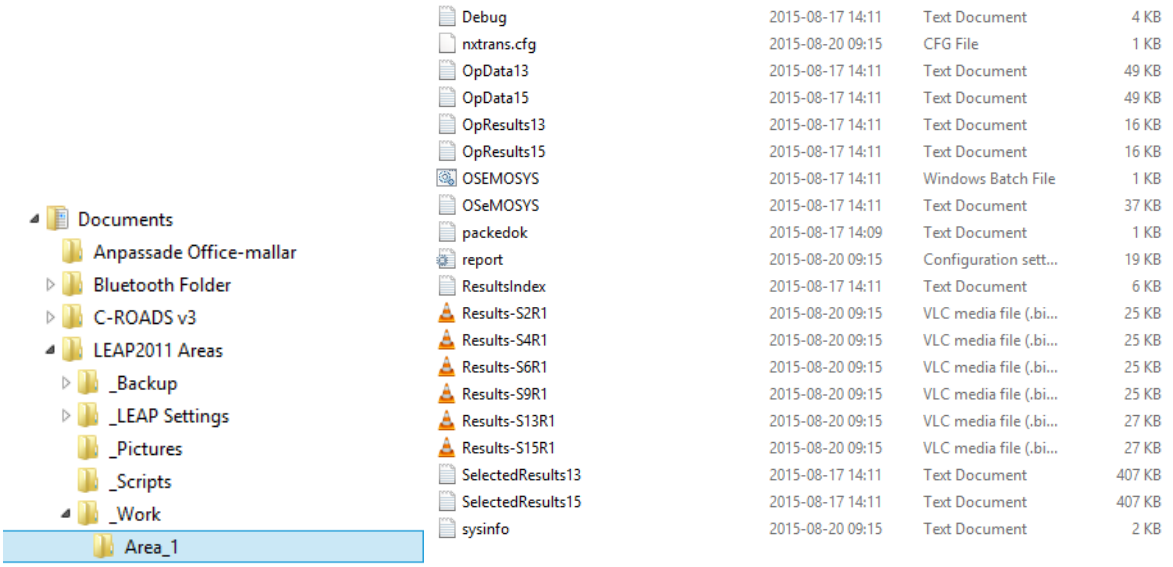

Figure 2. LEAP model file in Area_1

The OpData OSeMOSYS data file can then be opened in NotePad++, edited and used further within OSeMOSYS. Using LEAP to set up an OSeMOSYS data file ensures that the OSeMOSYS data file is set up correctly, thus making it easier to build the remaining RES for further optimisation in OSeMOSYS. This may for example be useful to optimise other non-electricity sectors, such as transport or heat, as this is currently not possible within LEAP. Further, if new equations should be added to an updated OSeMOSYS version (that is therefore not supported by LEAP), it may as well be useful to take the LEAP data file as a starting point.

3.5 DATA AND CHOICE OF UNITS

The cornerstone of a legitimate model is the data. Using accurate data, relevant model designs and a consistent choice of assumptions will ultimately offer better and more representative insights into the system. Typical data requirements include:

- **Energy demand** for the activities that are considered in the model and an **annual load curve** (for electricity, heating and cooling);
- **Efficiencies**, the **capacity** and the **capacity/availability factor** of the power production technologies;
- Corresponding **costs for technologies**;
- Constraints like **import constraints**, **resource levels** etc.;
- **Emissions** and others.

Useful technology briefs containing such data have been developed by ETSAP (http://www.etsap.org/). The World Energy Outlook from IEA (http://www.worldenergyoutlook.org/), IEA Cost of Generating Electricity (look for latest publication at www.iea.org) and IRENA's Renewable energy publications can further be used to obtain the required data for modelling a country's energy sector. The fossil fuel reserves in every country can be obtained from EIA (U.S- Energy Information Administration) [4]. Also the World Bank database is a useful source of data for energy demand. Note that these publications provide generic values, and national studies and strategy documents are usually preferable over generic, more global values.

For OSeMOSYS there are 4 units that needs to be chosen carefully. Bear in mind that certain default constraint levels, e.g., the total max capacity are set to values such as 999999. This limits can be violated if the choice of unit is too small (like kW for a large system, for which GW are recommendable).

	Possible choice of unit
Energy	GWh, MWh, PJ, GJ, etc.
Power	GW, MW, etc.
Cost	Million $, Million £, Million Euro, etc.
Emissions	Mton

There is **no unit conversion** in OSeMOSYS: the modelling system assumes that all units are consistent. For example, the unit for capital costs needs to be coherent with the choice of units from the above table and is applied for all parameters relating to costs. E.g., when choosing GW and $ as power and monetary units respectively, the **Capital cost** has to be defined in $/GW. Similarly, if the energy unit is PJ for the demand then it also needs to be chosen for all other parameters activity/generation/output related parameters. This is particularly important for the parameter **CapacityToActivityUnit** (see p.11). [2] [3].

3.6 SUPPORTIVE PROGRAMMES AND DOCUMENTATION

The installation of Notepad ++ is recommended to work with and edit the model and data files. It can be downloaded here [5].

The following three files are recommended if more background on the basics of GnuMathProg and the linear optimisation logic applied in OSeMOSYS are of interest.

1) http://www.osemosys.org/uploads/1/8/5/0/18504136/ceron_-_2006_-_the_gnu_linear_programming_kit,_part_1_-_introduction_to_linear_optimization.pdf

2) http://www.osemosys.org/uploads/1/8/5/0/18504136/ceron_-_2006_-_the_gnu_linear_programming_kit,_part_2_-_intermediate_problems_in_linear_programming.pdf

3) http://www.osemosys.org/uploads/1/8/5/0/18504136/ceron_-_2006_-_the_gnu_linear_programming_kit,_part_3_-_advanced_problems_and_elegant_solutions.pdf

Further, as mentioned before, the most comprehensive description of how OSeMOSYS works is provided in *"OSeMOSYS: The Open Source Energy Modeling System, An introduction to its ethos, structure and development" by M. Howells et al. in 2011 and "Modelling elements of Smart Grids – Enhancing the OSeMOSYS (Open Source Energy Modelling System) code" by* Welsch, M. et al. in 2012. It should be noted that the salvage value as described in Howells M. et al., 2011 is not applicable anymore please see the changelog provided at www.osemosys.org for latest changes. Further, the description of storage in Howells M. et al., 2011 is not applicable any longer. Instead, refer to the current way of modelling storage or variability as described in Welsch M. et al., 2012.

[redacted] PARAMETER DESCRIPTION

In this chapter the sets and parameters in OSeMOSYS are described. Illustrative examples taken from UTOPIA are also added to illustrate how the resulting input file could look like. It is therefore recommendable to read this section while comparing the text with the UTOPIA file, opened in Notepad ++.

This note sets out how the data file is structured to compile an energy model in OSeMOSYS. For easy reference, it is set out in the same structure and order as the inputs are specified in the OSeMOSYS code. It is also written in an informal style with comments aimed at helping the new user to become more familiar with the system. It is up to the analyst to use a consistent set of units as pointed out in Section 3.5.

4.1 SETS

First off the model sets need to be defined. These are the *elements* that will be modelled. Mathematically, they represent the indices within the equations of the model constraints. I.e., the constraints are formulated for all combinations of all (or a selection of) the sets outlined below.

These include[3]:

- The **EMISSION** to be accounted for. These are indexed in the mathematical formulations by the letter "e".
- The **TECHNOLOGY**s modelled. Note that this includes any element of the energy system which generates a fuel (e.g., a coal mine or a wind farm), converts one energy form into another (e.g., a coal-fired power plant), or consumes a fuel (e.g., an air conditioner). Technologies are indexed by the letter "t".
- The **FUELS** used. Energy carriers (fuels) required in the model have to be produced by a technology. Energy carriers (fuels) are only produced if they are going to be consumed or if they feed a final demand. Also demands for energy services are defined as fuels in OSeMOSYS, e.g., a heating demand would be defined as a fuel. All fuels are indexed by the letter "f".
- The **YEAR**s modelled, indexed by the letter "y".
- The **TIMESLICE**s, each of which combines a fraction of the year with specific load and supply characteristics. One time slice could, for example, represent all the weekend evenings in summer, another one the weekday evenings in winter. As such, time slices do not have an inherent chronology. The time slices are indexed by the letter "l".
- The number of **MODES_OF_OPERATION** that a technology can have, assuming that the technology can switch between a linear combination of these. Modes of operation are usually defined if a technology can use various input or output fuels and can choose the mix of these input or output fuels. For example, a CHP plant may vary between producing heat in one "mode of operation" and electricity in another. The "capacity" remains constant simply because the same piece of machinery produces both outputs. The modes of operation are indexed by the letter "m".
- The **REGIONS** that are to be modelled. Commonly, one country is modelled as one region and it may be efficient from a computational point of view to even multiple countries within a single region, e.g., by defining different fuels and technologies for each country. In this case, countries could be linked to one another by setting up technologies which link one country to another by converting the fuels used in one country to the fuels used in another. This has the advantage that only one region has to be defined and thus all equations are only defined for one region. Defining, e.g., two regions would roughly double the amount of equations in the model and thus also the computational burden. Regions are indexed by the letter "r".
- The **STORAGE** set contains the storage facilities. This is indexed by the letter "s".

[3] Note that all indexes are lower case.

- The **SEASON**s are indexed as "ls", **DAYTYPE**s as "ld" and **DAILYTIMEBRACKET**s as "lh" and are all connected to the TIMESLICE "l". They are only required if storage will be used in the model. Note that these three sets need to be defined as numerical, consecutive values, e.g., 1, 2, 3. Conversion factors are used when modelling storage to attribute each time slice to a specific season, day-type and daily time bracket (refer to Section 4.8). Seasons occur within a year, and day-types within a week, e.g., weekdays and weekends. Daily time brackets define the number of brackets that the days is divided in ,e.g., one hour, or mornings, afternoons and evenings. If no storage is modelled, it is sufficient to define each of these three sets with one numerical element, e.g., "1".

EXAMPLE **SETS** FROM UTOPIA

Below are the sets defined in the sample file UTOPIA. Note that abbreviations are used, e.g., for emissions, technologies and fuels. However, any other name could be used to describe those, as long as they are consistently used throughout the model.

```
set   EMISSION   :=   CO2   NOX;

set   TECHNOLOGY   :=   E01   E21   E31   E51
      E70   IMPDSL1   IMPGSL1   IMPHCO1   IMPOIL1   IMPURN1
      RHE   RHO   RL1   SRE   TXD   TXE
      TXG   RIV   RHu   RLu   TXu;

set   FUEL   :=   CSV   DSL   ELC   GSL
      HCO   HYD   LTH   OIL   URN   RH
      RL   TX;

set   YEAR   :=   1990   1991   1992   1993
      1994   1995   1996   1997   1998   1999
      2000   2001   2002   2003   2004   2005
      2006   2007   2008   2009   2010;

set   TIMESLICE   :=   ID   IN   SD   SN
      WD   WN;

set   MODE_OF_OPERATION   :=   1   2;

set   REGION   :=   UTOPIA;

set   SEASON   :=   1   2   3;

set   DAYTYPE   :=   1;

set   DAILYTIMEBRACKET   :=   1   2;

set   STORAGE   :=   DAM;
```

4.2 PARAMETERS

The parameters are entered next. They are the input values defined for certain combinations of sets, as outlined before. All parameters must be defined to be able to calculate the resulting variables. Variables constitute the output values from the model run.

4.3 SYNTAX FOR THE PARAMETERS

All parameters are entered in the form of a matrix, some of which have several dimensions. One default value can be defined for all elements of one specific parameter. If an element is explicitly assigned a diverging value in the matrix defining a parameter, this diverging value will replace the default value. When specifying a matrix with more than two dimensions, using a "*" instead of an element of a set will indicate which set is listed in the column (first "*" when listing the sets) and which set is listed in the row (second "*" when listing the sets). For example, the parameter AccumulatedAnnualDemand[r,f,y] listed below is defined by the sets [r,f,y], i.e., for each [REGION, FUEL, YEAR]. By default it is defined to be zero. I.e., if an accumulated annual demand would not be listed for a fuel in the matrix below, it would be zero for this fuel. The first "*" is in the position of the set FUEL, which is therefore defined in the row of the matrix (TX stands for transport fuel). The second "*" is in the position of the set YEAR, which are therefore listed in the columns. The region entered is UTOPIA, which is the only region in this case. If there were several regions, simply one region would be defined after another in several such matrices. Only the last data entry of the last matrix will be followed by the ";", which is always required to indicate the end of a parameter definition.

param	AccumulatedAnnualDemand	default	0:=		
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=
TX	5.2	5.46	5.72	5.98	6.24;

*Please note that in the two background papers by Howells, M. et al., 2011 and Welsch, M. et al., 2012 the parameters are **indexed differently** to the current version of the OSeMOSYS input file. Please refer to the Change log on www.osemosys.org for further background on this switch of indices.*

4.4 GLOBAL PARAMETERS

The first type of parameters are "Global" in that they define the global aspects of the model framework.

The parameter **YearSplit**[l,y] is a matrix which measures the duration of a modelled time slice as a fraction of the year. Model years (defined in the SETs) are in columns and timeslices (defined in the SETs) in the rows. The same values are used for all regions. The sum of each entry over the year should equal 1. Consider, for example, a model constructed with a year that has two TIMESLICES, "Summer" and "Winter" and those are the same length. Then the YearSplit parameter for each TIMESLICE in that year will be 0.5.

If you want to summarise all weekday mornings in summer you would, for example, name it SWM in the set **TIMESLICE** (be consistent) and attribute a year share of:

Summer months	**June**	**July**	**August**	**SUM**
Days per month	30	31	31	
Weekdays	22	23	21	=22+23+21 = **66 days**
Hours in the morning	6	6	6	=66 * 6 = **396 hours**
Hours per year				**8760 hours**
YearSplit for SWM				=396/8760 = **0.045**

This would be reflected in the YearSplit parameter as follows:

EXAMPLE FROM UTOPIA **YEARSPLIT**

param	YearSplit	:	1990	1991	1992	:=
ID			0.1667	0.1667	0.1667	
IN			0.0833	0.0833	0.0833	
SWM			**0.045**	**0.045**	**0.045**	
SD			0.205	0.205	0.205	
WD			0.3333	0.3333	0.3333	
WN			0.1667	0.1667	0.1667	

Make sure that the sum for each year equals 1!

The parameter **DaySplit**[lh,y] is the 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).

EXAMPLE **DAYSPLIT**

param DaySplit default 0.00137 :=;

The parameter **DaysInDayType[**ls,ld,y] represents the number of days for each day type within a week, i.e., out of 7.

EXAMPLE DAYSINDAYTYPE

param DaysInDayType default 7:=

The **DiscountRate**[r,t] parameter is a matrix that gives the discount rate for each technology in each region. Technologies are given in columns and regions in rows. The parameter is used to calculate a discount factor for the start of each year. That is used when discounting new capacity investments and any other costs such as operating costs. It is also used (together with the lifetime of a technology) to calculate the salvage value of each technology invested in during the model period, but whose life extends beyond the final model year.

In the example files provided, a default (and common) discount rate is used. Having separate discount rates can be useful when trying to simulate specific "hurdle" rates that differ from sector to sector. However, this is not possible in the version OSeMOSYS_2013_05_10 or OSeMOSYS_2013_05_10_short. **Therefore, only use a single default value as of now and check the change log for updated functionality.**

EXAMPLE OF **DISCOUNTRATE** UTOPIA

param DiscountRate default 0.05 :=;

4.5 DEMANDS

Next, the user enters the demands. Note that these demands are commonly defined for energy-services or fuels. There are two different kinds of demands. One is an *accumulated annual demand*. This type of demand can be satisfied at any time of the year, as long as the total is met. This is a simplification that is useful when storage is not a constraint (e.g., to enter a demand for oil when there are enough reserves available within a country). It should however not be used for modelling an electricity demand, as electricity has to be supplied

instantly when a demand occurs. Demands for energy carriers (i.e., fuels) such as electricity should rather be modelled as a *specified annual demand*, which has a specific demand profile during the year that must be satisfied.

The units are set by the user. However, note that these should be consistent for all demands. A suggested unit for a national model is PJ/a.

The **AccumulatedAnnualDemand[r,f,y]** parameter is entered as a matrix which can be specified for any fuel. A default demand is set to zero (so no annual demand is given to fuels not listed in the matrix). Model YEARs are in the column index and FUELs in the rows. This matrix is repeated for each REGION modelled.

EXAMPLE ACCUMULATEDANNUALDEMAND

param	AccumulatedAnnualDemand	default	0:=		
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=
TX	5.2	5.46	5.72	5.98	6.24;

There are demands for which time of use is necessarily specified during the year. These are included in the **SpecifiedAnnualDemand[r,f,y]** parameter. As the name implies, this parameter contains the total specified demand for the year. (This is distributed by timeslice with the next parameter.) The parameter is entered as a matrix which can be specified for each fuel/energy service demand. A default demand is set to zero (so no specified demand is given to fuels not listed in the matrix). Model YEARs are in the row index and FUELs in the columns. The matrix is repeated for each REGION modelled.

EXAMPLE OF SPECIFIEDANNUALDEMAND

param	SpecifiedAnnualDemand	default	0:=		
[UTOPIA,*,*] :	1990	1991	1992	1993	1994:=
RH	25.2	26.46	27.72	28.98	30.24;

For each energy-service or fuel which has a SpecifiedAnnualDemand, a demand profile should be assigned. This is included in the **SpecifiedDemandProfile**[r,f,l,y] parameter. (If this is left blank then no specified demand is calculated.) The SpecifiedDemandProfile parameter indicates the annual fraction of energy-service or fuel demand that is required in each time slice. For each year these should sum up to one.

EXAMPLE OF SPECIFIEDDEMANDPROFILE

param	SpecifiedDemandProfile	default	0:=		
[UTOPIA,**RH**,*,*]:	1990	1991	1992	1993	1994:=
ID	0.12	0.12	0.12	0.12	0.12
IN	0.06	0.06	0.06	0.06	0.06
SD	0	0	0	0	0

SN	0	0	0	0	0
WD	0.5467	0.5467	0.5467	0.5467	0.5467
WN	0.2733	0.2733	0.2733	0.2733	0.2733;

Make sure that the sum for each year equals 1!

4.6 TECHNOLOGY

In a departure from several existing modelling frameworks, all energy system components are set up as a technology in OSeMOSYS. For example, a resource is represented as a technology with no input fuel, but producing an output fuel with limits on its activity to represent the resource availability.

4.6.1 PERFORMANCE CHARACTERISTICS

As the capacity may be measured in one unit and energy in another, the parameter **CapacityToActivityUnit**[r,t] is specified for each technology and for each region. TECHNOLOGY is given in columns and REGION in rows. The parameter effectively relates the energy (in the chosen energy units) that would be produced were one unit of capacity (in the chosen capacity unit) fully used for one year. Thus if capacity is measured in GW and energy in PJ, for a given technology, then this parameter would have a value of 31.536.

EXAMPLE OF CAPACITYTOACTIVITYUNIT

param	CapacityToActivityUnit		default	1:		
		E01	E21	E31	E51	E7:=
UTOPIA		31.536	31.536	31.536	31.536	31.536;

In the parameter **TechWithCapacityNeededToMeetPeakTS**[r,t], **unlike the name suggests**, all technologies which operate timeslice dependent need to be tagged (i.e., set equal to 1). E.g., all power plants like a coal-fired power plant need to be tagged with this parameter. A coal mine however will not need to be tagged with this parameter, as it may mine coal constantly throughout the year, independently of the actual timeslice-dependent demand of the coal-fired power plant. This is because storage for coal is cheap and available.

If the value of this parameter is 1, equations are looked at on a timeslice bases (such as for power plants). If the value = 0 (or any number different to 1), then the equations are just calculated on yearly bases (such as for a coal mine).

TECHNOLOGY is given in columns and REGION in rows.

EXAMPLE FOR TECHWITHCAPACITYNEEDEDTOMEETPEAKTS

param	TechWithCapacityNeededToMeetPeakTS			default	0:	
	E01	E21	E31	E51	E70	
UTOPIA	1	1	1	1	1;	

The parameter **InputActivityRatio**[r,t,f,m,y] gives the rate of input (use) of fuel for a technology as a ratio to the rate of activity. Commonly, it is advisable to define the InputActivityRatio as the inverse efficiency (η^{-1}=) of the technology and set the OutputActivityRatio (described later in this document) equal to one. Like this, the

rate of activity of a technology equals the output of a technology. E.g., if a power plant is active at 87 MW it produces 87 MWh per hour, while consuming 87 MWh * η^{-1} of input fuel per hour. Further, all cost parameters, which are related to the rate of activity of a technology, then refer to the output of the technology. This makes sense for all power plants, as, e.g., capacity costs are commonly given per unit of output.

However, the capacity of refineries may be measured in terms of barrels of crude oil processed per day, i.e., per fuel input. In such cases it is preferable to set the InputActivityRatio equal to one and enter the efficiency in the OutputActivityRatio. Like this, the activity of the refinery equals the input to this technology. E.g., if the oil refinery is active at a certain level, this level corresponds to the amount of input fuel processed to produce this level times η of output fuel (e.g., gasoline). Further, all cost parameters are then related to this fuel input, i.e., the barrels of crude oil processed (using the units of energy chosen upfront for all technologies by the analyst, e.g., PJ).

The InputActivityRatio is defined by year (as the efficiency of the technology could degrade), technology, fuel produced, the mode of operation and region. Care needs to be taken to enter this matrix correctly because of its multi-dimensional form. It is suggested that, for each fuel and mode of operation, a two dimensional matrix (with years in the columns and technologies in the rows) be constructed.

EXAMPLE INPUTACTIVITYRATIO

param	InputActivityRatio		default	0:=	
[UTOPIA,*,DSL,1,*]:	1990	1991	1992	1993	1994:=
E70	3.4	3.4	3.4	3.4	3.4
RHO	1.428571429	1.428571429	1.428571429	1.428571429	1.428571429
TXD	1	1	1	1	1
[UTOPIA,*,ELC,1,*]:	1990	1991	1992	1993	1994:=
RHE	1	1	1	1	1
RL1	1	1	1	1	1
TXE	1	1	1	1	1;

The parameter **OutputActivityRatio**[y,t,f,m,r] gives the rate of output of fuel as a ratio to the rate of activity in which a technology is operating. Please bear in mind that the InputActivityRatio and OutputActivityRatio are connected and it is preferable to place the inverse efficiency (η^{-1}) at the InputActivityRatio and keep the OutputActivityRatio at 1 for power production. It is defined by year (as the efficiency of the technology could degrade), technology, fuel produced, technology, mode of operation as well as by region. Care needs to be taken to enter this matrix correctly because of its multi-dimensional form. It is suggested that, for each fuel and mode of operation, a two dimensional matrix (with years in the columns and technologies in the rows) be constructed. Note also that if the variable and capacity cost parameters are being entered in terms of the technology's output, then the OutputActivityRatio should be given the value "1". (The latter representation is useful for power stations, who's capacity is reported in terms of electricity power output.)

EXAMPLE OUTPUTACTIVITYRATIO

param	OutputActivityRatio		default	0:=

[UTOPIA,*,RH,1,*]:	1990	1991	1992	1993	1994:=
RHE	1	1	1	1	1
RHO	1	1	1	1	1
RHu	1	1	1	1	1
[UTOPIA,*,RL,1,*]:	1990	1991	1992	1993	1994:=
RL1	1	1	1	1	1
RLu	1	1	1	1	1;

The fixed cost parameter **FixedCost**[r,t,y] is defined for each year, each technology and each region. It is the cost per unit of capacity of that technology. As not all technologies (as represented in OSEMOSYS) incur a fixed cost, this is left as zero by default. It is represented as a matrix with model years as the column index and the technologies as row index. The unit is currency per unit of capacity.

EXAMPLE OF **FIXEDCOST**

param	FixedCost	default	0:=			
[UTOPIA,*,*]	:	1990	1991	1992	1993	1994:=
E01		40	40	40	40	40
E21		500	500	500	500	500;

The **CapitalCost**[r,t,y] of each technology is given as a function of the technology as well as the year in which the technology is invested in. Note that OSeMOSYS does not include a construction time period. Any interest occurred during the construction time therefore needs to be accounted for externally when entering this value. The user should take care to ensure that this is consistent with the discount rate used.

Not all technologies (as represented in OSeMOSYS) may incur a capacity cost. (E.g., a technology may represent the sun and its availability, which is then used by other technologies like solar panels). The default value is therefore zero. It is suggested that data is entered two dimensional matrix with model years in the columns and technologies in the rows. The units are in currency per unit of installed capacity.

EXAMPLE CAPITALCOST

param	CapitalCost	default	0:=		
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=
E01	1400	1390	1380	1370	1360
E21	5000	5000	5000	5000	5000;

The **VariableCost** [r,t,m,y] is defined for each year, technology, mode of operation and region. It is the cost per unit of activity (for a given mode of operation) of that technology. As not all technologies (as represented in OSeMOSYS) incur a variable cost, this is set to zero by default. It is suggested that the data be entered as a two

dimensional matrix for each mode of operation that the technology has. In this form, model years are the column index and the technologies in the rows.

EXAMPLE OF VARIABLECOST

param	VariableCost	default	0:=			
[UTOPIA,*,1,*]	:	1990	1991	1992	1993	1994:=
E01		0.3	0.3	0.3	0.3	0.3
E21		1.5	1.5	1.5	1.5	1.5;

ResidualCapacity[r,t,y] is entered for each model year, technology and each region. It is the capacity left over from a period prior to the modelling period. Unlike MESSAGE, LEAP and several other modelling frameworks, it is not calculated automatically based on historical investments and a lifetime associated with those investments. Therefore this needs to be calculated outside of OSeMOSYS manually and added to the ResidualCapacity parameter. I.e., the value for a technology should decrease over time as historic capacity is retired. Model years are given in the column index and the technologies in the rows. The units are in capacity.

EXAMPLE OF RESIDUALCAPACITY

param	ResidualCapacity		default	0:=		
[UTOPIA,*,*]	:	1990	1991	1992	1993	1994:=
E01		0.5	0.5	0.5	0.4	0.4
E21		0	0	0	0	0;

The **AvailabilityFactor**[r,t,y] parameter is defined for each model region, technology and year. It is often used to simulate "planned outages" and indicates the maximum time a technology may run for the whole year. The model will then decide by itself in which timeslices the output will be reduced if the value given is lower than 1, which constitutes its default value. E.g., if there would be ten timeslices, a constant capacity factor and an availability factor of 0.9, the model may choose to operate the technology up to is capacity factor in all but one timeslice, in which the technology will not produce anything. Alternatively, it may choose to operate it in all but two timeslices, in which the technology only produces half of what the capacity factor would allow. It is not recommended to combine an availability factor smaller then 1 with capacity factors varying over the year[4]. Model years are given in the columns and technologies in the rows.

EXAMPLE OF AVAILABILITYFACTOR

param	AvailabilityFactor		default	1:=	
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=

[4] The actual output of a technology in OSeMOSYS is calculated as the capacity factor in each time slice summed up over a year, with is then multiplied by the availability factor. E.g., if a technology has a capacity factor which is zero in five timeslices and one in five other timeslices and an availability factor of 0.9, it could only produce 50% times 0.9 = 45% of its total capacity throughout the year. Yet, if the availability factor would represent maintenance work, one would schedule this maintenance work rather in those timeslices where the technology is not producing, resulting in a yearly output of 50% instead of 45%.

RHE	1	1	1	1	1
RHO	1	1	1	1	1;

The **CapacityFactor**[r,t,l,y] is defined for each region, technology, timeslice and year. It is used to convert annual capacity to the capacity available for each timeslice, i.e., the actual energy output vs. the potential energy output at the maximum capacity of a technology. By default it is given the value 1 (meaning that all of the capacity is available for each timeslice). Model years are given in columns and the timeslice in the rows. This is repeated for each technology. Only technologies for which there is a capacity factor (to mimic "forced outage" or to de-rate them, e.g., to model wind availabilities in different times of the year) need explicitly be included. If a PV panel should be modelled then the daytime capacity factor could be 0.4 during the day and 0 during the night time with an average of 0.2 over the year.

EXAMPLE OF CAPACITYFACTOR

param	CapacityFactor	default	1:=			
[UTOPIA,E01,*,*]:		1990	1991	1992	1993	1994:=
ID		0.8	0.8	0.8	0.8	0.8
IN		0.8	0.8	0.8	0.8	0.8;

Each technology is given a limited operational lifespan. This is included in the OperationalLife[r,t], presented as a matrix with technologies in columns and regions in rows.

EXAMPLE OF OPERATIONALLIFE

param	OperationalLife	default	1:			
E01	E21	E31	E51	E70	RHE	RHO:=
UTOPIA	40	40	100	100	40	30;

4.6.2 CAPACITY LIMITS

If the sum of all technology investments are allowed up to a maximum total (**residual, i.e., historic, and total new**) capacity each year of the modelling period, the limiting capacity level is included in the **TotalAnnualMaxCapacity**[r,t,y] parameter. The default value for all technologies is a high number (so that the limit is usually not enforced). The unit of the parameter is the unit that is used to measure the respective technologies' capacity in the model. The parameter is a matrix with the years as the columns and technologies as the rows index.

EXAMPLE OF TOTALANNUALMAXCAPACITY

param	TotalAnnualMaxCapacity		default	99999:=	
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=
E31	0.1301	0.1401	0.1401	0.1501	0.1501

E51	3	3	3	3	3;

If a technology should be invested in to reach a minimum total (residual and new) capacity each year, the limiting capacity level is included in the **TotalAnnualMinCapacity**[r,t,y] parameter. The default for all technologies is zero (so that the limit is effectively not enforced). The unit of the parameter is the unit that is used to measure the respective technologies' capacity in the model. The parameter is a matrix with the years as the columns and technologies in the rows.

EXAMPLE OF TOTALANNUALMINCAPACITY

param	TotalAnnualMinCapacity	default	0	:=	
[UTOPIA,*,*]	:	1990	1991	1992	1993	1994:=
E31		0.13	0.14	0.14	0.15	0.15
SRE		0.1	0.1	0.1	0.1	0.1;

If annual investments in new capacities for a given technology are limited to a given level each year, this upper limit is included in the **TotalAnnualMaxCapacityInvestment**[r,t,y] parameter. The default for all technologies is a high number (so that the limit is usually not enforced). The unit of the parameter is the unit that is used to measure the respective technologies capacity in the model. The parameter is a matrix with the years as the rows and technologies included in the columns.

EXAMPLE OF TOTALANNUALMAXCAPACITYINVESTMENT

param	TotalAnnualMaxCapacityInvestment	default	99999:=;

If annual investments in new capacities for a given technology are limited to a given level each year, this lower limit is included in the **TotalAnnualMinCapacityInvestment**[r,t,y] parameter. The default for all technologies is zero (so that the limit is effectively not enforced). The unit of the parameter is the unit that is used to measure the respective technologies capacity in the model. The parameter is a matrix with the years as the columns and technologies in the rows.

EXAMPLE OF TOTALANNUALMINCAPACITYINVESTMENT

param	TotalAnnualMinCapacityInvestment	default	0:=;

4.6.3 ACTIVITY LIMITS

It may be the case that there is an annual limit to the rate of activity, summed up over all "modes of operation", i.e., the total amount a technology could produce each year. If that is the case, the **TotalTechnologyAnnualActivityUpperLimit**[r,t,y] parameter should be used. The default value of this parameter is high (so that it is usually not enforced). The unit of the parameter is the unit that is used to measure the respective technology's rate of activity in the model. The parameter is a matrix with the years as the columns and technologies in the rows.

EXAMPLE OF TOTALTECHNOLOGYANNUALACTIVITYUPPERLIMIT

param	TotalTechnologyAnnualActivityUpperLimit	default	99999:=;

It may be the case that there is minimum rate of activity that a technology (considering the sum of all "modes of operation") must produce each year. If that is the case, the **TotalTechnologyAnnualActivityLowerLimit**[r,t,y] parameter should be invoked. The default value of this parameter is zero (so that it is effectively not enforced). The unit of the parameter is the unit that is used to measure the respective technologies activity in the model. The parameter is a matrix with the years as columns and technologies in the rows.

EXAMPLE OF TOTALTECHNOLOGYANNUALACTIVITYLOWERLIMIT

param	TotalTechnologyAnnualActivityLowerLimit	default	0:=;

Should there be a maximum level of activity by a technology (summed over all modes of activity) over the whole model period, the parameter **TotalTechnologyModelPeriodActivityUpperLimit**[r,t,y] should be used. The default value of this parameter is high (so that it is usually not enforced). The unit of the parameter is the unit that is used to measure the respective technologies activity in the model.

EXAMPLE OF TOTALTECHNOLOGYMODELPERIODACTIVITYUPPERLIMIT

param	TotalTechnologyModelPeriodActivityUpperLimit	default	99999:=;

Should there be a minimum level of activity (summed over all modes of activity) to be enforced by a technology over the whole model period, the parameter **TotalTechnologyModelPeriodActivityLowerLimit**[r,t] should be used. The default value of this parameter is zero (so that it is effectively not enforced). The unit of the parameter is the unit that is used to measure the respective technologies activity in the model.

EXAMPLE OF TOTALTECHNOLOGYMODELPERIODACTIVITYLOWERLIMIT

param	TotalTechnologyModelPeriodActivityLowerLimit	default	0:=;

4.7 EMISSIONS

The parameter **EmissionActivityRatio** [r,t,e,m,y] gives the emission levels per quantity fuel of a particular mode of activity for a technology. It is defined by year, technology, emission, as well as by the technologies' mode of operation. Care needs to be taken to enter this matrix correctly, due to its multi-dimensional form. The units are the units of the emissions per unit of energy, e.g., kton per PJ. As this value is multiplied by the rate of activity, it is important to be aware if the rate of activity equals the input fuel consumed or the output fuel consumes, i.e., which of these values has been set to one when defining the InputActivityRatio and OutputActivityRatio.

EXAMPLE OF EMISSIONACTIVITYRATIO

param	EmissionActivityRatio	default	0:=;		
[UTOPIA,*,CO2,1,*]:	1990	1991	1992	1993	1994:=
IMPDSL1	0.075	0.075	0.075	0.075	0.075
IMPGSL1	0.075	0.075	0.075	0.075	0.075;

An **EmissionsPenalty** [r,e,y] is defined for each region, emission and year . It is represented as a matrix with model years as the columns and emissions as rows. It is measures in the units of costs per unit of activity, e.g., Million EUR/PJ.

EXAMPLE OF EMISSIONSPENALTY

```
param           EmissionsPenalty:=

[UTOPIA,*,*]    :       1990    1991    1992    1993    1994:=

CO2                     0       0       0       0       0;
```

If there are emissions that need to be accounted for, but are not calculated by the model on an annual basis, they can be included as annual exogenous emissions for each year and region in the parameter **AnnualExogenousEmission**[r,e,y]. The parameter is entered as a matrix with model years as columns and emissions (in emissions units) in the rows.

EXAMPLE OF ANNUALEXOGENOUSEMISSION

```
param           AnnualExogenousEmission     default     0:=;
```

The sum of all emissions from the energy system being modelled (plus any annual exogenous emissions) can be limited using the **AnnualEmissionLimit**[r,e,y] parameter. It is entered as a matrix with model years as the columns and emissions (in emissions units) in the rows.

EXAMPLE OF ANNUALEMISSIONLIMIT

```
param           AnnualEmissionLimit     default     9999:=;
```

If there are emissions that need to be accounted for, but are not calculated by the model over the entire model period, they can be included as annual exogenous emissions for each year and region in the parameter **ModelPeriodExogenousEmission**[r,e]. The parameter is entered as a matrix with exogenous emissions as the columns and regions in the rows.

EXAMPLE OF MODELPERIODEXOGENOUSEMISSION

```
param           ModelPeriodExogenousEmission        default     0:=;
```

The sum of emissions from the energy system modelled over the model period (plus any annual or model period exogenous emissions) may be limited using the parameter **ModelPeriodEmissionLimit**[r,e]. The parameter is entered as a matrix with emissions (in emissions units) as the columns and regions in the rows.

EXAMPLE OF MODELPERIODEMISSIONLIMIT

```
param           ModelPeriodEmissionLimit        default     9999:=      ;
```

4.8 RESERVE MARGIN

Fuel(s) that require a reserve margin are given a value of "1" in the **ReserveMarginTagFuel**[r,f,y] parameter. The default otherwise is zero. The parameter is a matrix with the years in the columns and fuels included in the rows.

EXAMPLE OF RESERVEMARGINTAGFUEL

param	ReserveMarginTagFuel		default	0:=	
[UTOPIA,*,*]:	1990	1991	1992	1993	1994:=
ELC	1	1	1	1	1;

The **ReserveMargin**[r,y] parameter is given per year and indicates the reserve level of installed capacity required over the peak demand (as modelled) for the corresponding fuel. Note that only technologies that are tagged are included in this reserve margin. It is entered as a matrix with the years as the columns and regions included in the rows. The reserve margin is given as a unitless fraction. In the following case, 18% more total installed capacities are required then the actual peak load. (Note that the capacity factor is not applied to calculate the reserve requirements, i.e., the full installed capacity is taken).

EXAMPLE OF RESERVEMARGIN

param	ReserveMargin:	1990	1991	1992	1993	1994:=
UTOPIA		1.18	1.18	1.18	1.18	1.18;

Each technology (in each region) that is allowed to contribute to the reserve margin is tagged using the **ReserveMarginTagTechnology**[r,t,y]. If the "tag value" is 1, then 100% of the installed capacity of that technology contributes to the reserve. If the tag value is .2, then only 20%. This representation is useful, as, e.g., some variable renewable technologies contribute to the capacity reserve in a limited manner. The parameter is a matrix with the years as the columns and technologies included in the rows.

EXAMPLE OF RESERVEMARGINTAGTECHNOLOGY

param	ReserveMarginTagTechnology	default	0:=			
[UTOPIA,*,*]	:	1990	1991	1992	1993	1994:=
E01		1	1	1	1	1
E21		1	1	1	1	1;

4.9 STORAGE

How to model storage in OSeMOSYS is best described in Welsch, M., et al. in 2012. The following just provides a brief outline how to provide unlimited storage in the model, which corresponds to the UTOPIA case applied in this illustration. If there is no need to consider storage, simply do not link any technology to the storage site by setting the TechnologyToStorage and TechnologyFromStorage equal to zero.

Unlike in a more basic data file set up, chronological information is required when modelling storage levels. If storage is used then each time slice needs to be assigned to a season, day type and daily time bracket, as defined in the following three parameters.

The parameter **Conversionls**[l,ls] is set equal to 1 to assign a particular time slice to a season. Set it equal to 0 in order not to assign a particular time slice to a season. The parameter is a matrix with the season as columns and timeslice in the rows. In the example below for example, the third and fourth timeslices (SD = Summer Day and SN = Summer Night) are combined in one season called "3".

EXAMPLE CONVERSIONLS

param Conversionls default 0 :=

[*,*]:	1	2	3	:=
ID	0	1	0	
IN	0	1	0	
SD	0	0	1	
SN	0	0	1	
WD	1	0	0	
WN	1	0	0	;

The parameter **Conversionld**[l,ld] s set equal to 1 to assign a particular time slice to a day type. Set it equal to 0 in order not to assign a particular time slice to a day type. The set DAYTYPE is in the column index and TIMESLICE in the row index. In the following example there is only one day type in a season (e.g., there could be a differentiation between weekdays and weekends), to which all timeslices are assigned to.

EXAMPLE CONVERSIONLD

param	Conversionld	default	0	:=
[*,*]:		1	:=	
ID		1		
IN		1		
SD		1		
SN		1		
WD		1		
WN		1	;	

The parameter **Conversionlh[**l,lh] is set equal to 1 to assign a particular time slice to a daily time bracket. Set it equal to 0 in order not to assign a particular time slice to a daily time bracket. The DAILYTIMEBRACKET is in the

column index and TIMESLICE in the row index. In the following example, there are two DAILYTIMEBRACKETs in a given day type in a given season. E.g., SD and SN was assigned to one season before, and first comes the day (SD), which has the value 1, and then comes the night (SN) which has the value 2.

EXAMPLE CONVERSIONLH

param	Conversionlh	default	0	:=
[*,*]:		1	2	:=
ID		1	0	
IN		0	1	
SD		1	0	
SN		0	1	
WD		1	0	
WN		0	1	;

The parameter **TechnologyToStorage**[r,t,s,m] links a technology to a storage facility for charging the storage by assigning a value of 1. E.g., a pumped storage hydro power plant may be linked in the mode of operation "pump = 2" to a reservoir called "dam". The storage is given in the columns and technology in the rows for each mode of operation.

EXAMPLE OF TECHNOLOGYTOSTORAGE

param	TechnologyToStorage	default	0:=
[UTOPIA,*,*,2] :	DAM	:=	
E51	1	;	

The parameter **TechnologyFromStorage**[r,t,s,m] links a technology to a storage facility for discharging the storage by assigning a value of 1. E.g., a pumped storage hydro power plant may be linked in the mode of operation "turbine = 1" to a reservoir called "dam". The storage is given in the columns and technology in the rows for each mode of operation.

EXAMPLE TECHNOLOGYFROMSTORAGE

param	TechnologyFromStorage	default	0	:=
[UTOPIA,*,*,1]	:	DAM	:=	
E51		1	;	

The parameter **StorageLevelStart** sets the storage level at the beginning of first year in the units of energy available in the storage. Note: if it is zero, OSeMOSYS will use the first time slices in the entire first day type in the entire first season to fill the storage. To avoid OSeMOSYS taking a whole part of a season to fill up the

storage, and to avoid having to define shorter seasons, set it to zero, run the model, and check the StorageLevelYearStart variable of the following year and use a similar value for StorageLevelStart. Alternatively, model a few years before your first year interest.

EXAMPLE STORAGELEVELSTART

param StorageLevelStart default 0:=;

The parameter **StorageMaxChargeRate**{r,s] states the maximum charge that the storage can store (in units of power).

EXAMPLE STORAGEMAXCHARGERATE

param StorageMaxChargeRate default 99:=;

The parameter **StorageMaxDischargeRate**{r,s] is the rate that the storage can be discharged at (in units of power).

EXAMPLE STORAGEMAXDISCHARGERATE

param StorageMaxDischargeRate default 99:=;

The parameter **MinStorageCharge**[r,s,y] is given as a fraction of the maximum available storage level, i.e., between 0.00 and 0.99. The storage facility cannot be empties below this level.

EXAMPLE MINSTORAGECHARGE

param MinStorageCharge default 0. := ;

The parameter **OperationalLifeStorage**[r,s,y] is the storage's operational lifetime in years.

param OperationalLifeStorage default 99:= ;

The **CapitalCostStorage**[r,s,y] is the cost to invest in new storage facilities, defined per region, storage type and year. It is given in units of costs per units of energy, e.g., Million EUR/PJ

param CapitalCostStorage default 0:= ;

The parameter **DiscountRateStorage**[r,s] is the applied discount rate for the chosen storage per region, entered as a fraction, e.g., 1.05.

param DiscountRateStorage default 0:= ;

The parameter **ResidualStorageCapacity**[r,s,y] is the storage capacity which is available from before the modelling period, or which is known to become available in a specific year. It is given in the units of energy.

EXAMPLE RESIDUALSTORAGECAPACITY

```
param ResidualStorageCapacity default        999          :=
```

4.10 RENEWABLE ENERGY TARGETS

Each renewable technology may be tagged using the **RETagTechnology**[r,t,y] parameter. The "tag value" is either 1 for a renewable technology, or 0 otherwise. The parameter is a matrix with the years as the columns and technologies in the rows index. It is repeated for each region.

EXAMPLE OF RETAGTECHNOLOGY

```
param        RETagTechnology          default        0:=;
```

Similarly, the fuels (in each region) for which there is a renewable target are tagged using the **RETagFuel**[r,f,y] parameter.

EXAMPLE OF **RETAGFUEL**

```
param        RETagFuel       default        0:=;
```

The **REMinProductionTarget**[r,y] parameter indicates how much of the fuels (tagged in the RETagFuel parameter) must come from RE technologies (that were tagged with the RETechnology[y,t,r]. parameter). A value of 0 indicates that there is no RE target. The target is specified for each year in the units of energy.

EXAMPLE OF REMINPRODUCTIONTARGET

```
param        REMinProductionTarget          default        0            :=;
```

DEBUGGING THE MODEL

5.1 ADDING A DUMMY TECHNOLOGY

There can be several reasons why a model has "no feasible solution". One way of finding where the errors might be located is to add a dummy technology (which is not originally part of the model) that has high capacity and/or a very high variable cost and – to make things simple – no input fuel. This ensures that the model only uses the dummy technology when no other option remains. To debug a model, therefore define a single dummy technology and run the model to see if there is now a solution and start as close to the demand as possible, referred to as "1" in the below figure. If the model now runs successfully, check the results file and check when the dummy technology is used (e.g., only in certain time slices? Only in certain years?) and also check which other technologies are not used to the extent one would expect. This may give some clues where the error in the model is situated. After trying to find and fix the error, rerun the model and remove the dummy technology if it is not used any longer (alternatively, check every single time you run the model that the dummy technology is not used again, potentially for another reason if the model was modified in the meantime).

In more complex models, the model might solve after adding a dummy technology, yet it may still be unclear where the mistake in the model file might be. In these cases, revert back to the original model and add a

dummy technology in “2” in the RES (as seen in Figure 1) (and subsequently, if necessary, in “3”). This method will help to identify more clearly in which part of the Reference Energy System to mistake might be.

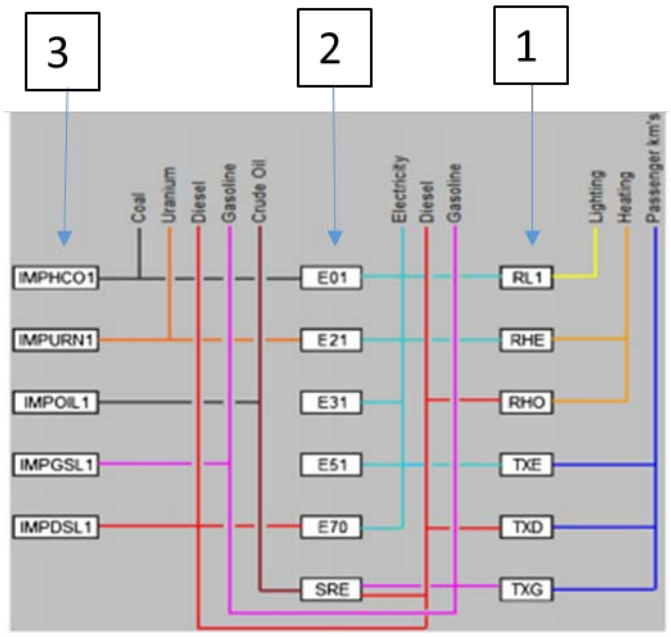

Figure 1. Strategy to add dummy technology in the RES.

5.2 UPPERLIMIT AND MAXLIMIT - INCREASE

If the optimisation has no feasible solution then one issue can be that certain upper limits or maximum limits have become binding and prevent the solver from finding a feasible solution. There are upper limits defined by the user but also default upper limits on parameters that are embedded in the code for OSeMOSYS. To find these “upperlimit” and “maxlimit” search in the code (Ctrl+F) and take a step by step approach both in the input file and OSeMOSYS code to identify if the issue is in one of these constraints to get a feasible solution. One option is for example to write a hash “#” before one or several of these constraints, thus commenting them out. If the model then solves successfully, it is clear that these constraints (potentially in combination with others) where the reason for the previous infeasibility and adjustments in the data file are necessary. These could involve correcting incorrectly entered input data or raising the maximum limits defined in the data file. Of course, the commented out constraints would have to be activated again in the corrected model by removing the hash “#” sign.

ADVANCED USES OF OSEMOSYS

6.1 USING OSEMOSYS WITH CPLEX

CPLEX is a commercial solver that is more powerful than the freely available GLPK solver. It is freely available for use by universities and in non-commercial projects. Especially when models get very complex, GLPK might solve very slowly. In these cases, other solvers such as CPLEX or GUROBI might be valuable alternatives. Below there is a quick summary of how to run OSeMOSYS using CPLEX.

1. In order to use CPLEX, the OSeMOSYS model and data files first need to be combined into a single .lp file. To do this, open the command prompt and use the following command: glpsol -m [OSeMOSYS model file] -d [Data file] --wlp [Input_Filename.lp]

2. After the lp file generation is done, close the command prompt window. Open CPLEX. Type 'read' and press enter. The name of the file to read is: '**C:\Input_Filepath\Input_Filename.lp**'

3. After the file is read, type '**optimize**' and press enter.

4. After the optimization is done, type 'write' and press enter. The name of file to write is: '**C:\Output_Filepath\Output_Filename.sol**'

5. Now that the solution file is written, close the CPLEX window and open the command prompt again. The solution file will need to be sorted and reordered. For this download the python sorting script that was developed for this function and copy it in to the Python 3.5.0 installation folder ('**C:\Python35**' is the default installation directory) . This script can be downloaded from the OSeMOSYS website. Go to the directory '**C:\Python35**'. Type **python transform_31072013.py Output_Filename.sol Output_Filename.txt**. For this step, you will need to have Python 3.5.0 installed. Python 3.5.0 can be downloaded from here. Click on the link "Download Python 3.5.0" and follow the installation instructions. The steps described here assume that the Python installation directory is **C:\Python35**.

6. After the python script is done, type '**sort/+1<C:\Python33\Output_Filepath\Output_Filename.txt>C:\Python33\Output_Filename_sorted.txt**'

The file produced through the above process contains the results of the model run in a format that is easy to analyse, either directly or after being copied into another software platform such as MS Excel.

6.2 MODIFICATIONS/ENHANCEMENTS/ADDITIONS TO THE MAIN CODE

If a user develops model code extensions, it is very much appreciated if these are communicated to the OSeMOSYS team (refer to the contacts given at www.osemosys.org). Code extensions are in general kept separate from the main code published on the website. When some enhancement or addition is proposed, it is advised to create an independent .txt or .doc file where the new parameters, variables and constraints are written in a format compatible with the main code. Any modification to existing constraints should be properly indicated. Further, a second file describing the modifications should be attached: in line with the concept of OSeMOSYS, this file should describe the modifications on four levels:

1) a conceptual description

2) a mathematical formulation

3) a formulation in GNU MathProg language, ready to be copied and pasted directly in the main code

4) Further, a sample file should be provided to test the model runs.

For an example of such modifications and their description please refer to: http://onlinelibrary.wiley.com/doi/10.1002/er.3250/abstract.

REFERENCE LIST

[1] OSeMOSYS Development team, “OSeMOSYS - Start up guide,” 06 07 2015. [Online]. Available: http://www.energycommunity.org/documents/OSeMOSYS/readme.pdf. [Accessed 06 07 2015].

[2] M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian and A. Roehrl, “OSeMOSYS: The Open Source Energy Modeling System, An introduction to its ethos, structure and development,” *Energy Policy,* vol. 39, p. 5850–5870, 2011.

[3] M. Welsch, M. Howells, M. Bazilian, J. DeCarolis, S. Hermann and H. Rogner, “Modelling elements of Smart Grids – Enhancing the OSeMOSYS (Open Source Energy Modelling System) code,” *Energy,* vol. Volume 46, no. Issue 1, p. Pages 337–350, October 2012.

[4] A. Gupta, A. Kartavtseva and G. Avgerinopoulos, “Training Manual – ANSWER OSeMOSYS,” KTH, Stockholm, 2013.

[5] OSeMOSYS Development team, “OSEeMOSYS - Getting started,” 06 07 2015. [Online]. Available: http://osemosysmain.yolasite.com/getting-started.php. [Accessed 06 07 2015].

[6] E. Van der Voort, “The EFOM 12C energy supply model within the EC,” *Omega,* vol. 10, no. 5, p. 507–523, 1982.

[7] Enerdata, “POLES: Prospective Outlook on Long Term Energy Systems,” 2015. [Online]. Available: http://www.enerdata.net/enerdatauk/solutions/energy-models/poles-model.php.

[8] ETSAP (Energy Technology Systems Analysis Program), Software and Tools, 2015. [Online]. Available: http://www.etsap.org/Tools.asp.

[9] IAEA (International Atomic Energy Agency), PESS Energy Models, 2015. [Online]. Available: https://www.iaea.org/OurWork/ST/NE/Pess/PESSenergymodels.html.

[10] NTUA (National Technical University of Athens), The PRIMES Energy, 2015. [Online]. Available: http://www.e3mlab.ntua.gr/manuals/PRIMsd.pdf.