ResLoadSIM

Version 3.3.0

Christoph Troyer

December 2018

Content

1 Installation

1.1 Requirements

Compiler

The most basic requirement is a C++ compiler. On a Linux box it is very likely that the GNU Compiler Collection (GCC) is already installed. If not, get it from there: http://gcc.gnu.org

If you need a free C++ compiler for a Windows machine, then look here:

http://www.mingw.org,

or get Visual C++ Express from Microsoft, which is also free:
http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express
Users of Mac OS X get a free development environment called Xcode. It can be downloaded from Apple's App Store.

GNU Scientific Library

The GSL provides a linear solver, which is used by resLoadSIM to calculate the space heating demand of households. It can be obtained from here: https://www.gnu.org/software/gsl

CMake (optional)

Cmake is a set of tools that help to manage the build process of a software package. It isn't a requirement for building ResLoadSIM. If you like to setup your makefiles by hand for example, or use an IDE like Xcode, which uses its own build system, then you can bypass Cmake, otherwise get it from here: http://www.cmake.org

The following description of the compilation process assumes that you use CMake though.

MPI (optional)

To compile and execute the parallel version of ResLoadSIM you need to install an implementation of the Message Passing Interface (MPI). One possibility is Open MPI, which can be found here:
http://www.open-mpi.org

PETSc (optional)

If ResLoadSIM is intended to be used with grid voltage control enabled, the PETSc package needs to be installed. It can be obtained via git:

```
git clone https://bitbucket.org/petsc/petsc.git
```

PETSc contains a power flow solver named pflow. The pflow code has been

modified by Shrirang Abhyankar to make it work together with ResLoadSIM. This updated version of pflow is available at:
https://github.com/abhyshr/pflow.git .

1.2 Compiling and Linking

The ResLoadSIM package comes as a compressed archive (*.tar.gz). The first thing you have to do is to uncompress the archive and extract the source code by executing the following command on the command-line (assuming you are using a Unix-like operating system):

```
tar xvfz resLoadSIM-x.y.z.tar.gz
```

Under Windows or Mac OS X you will have programs, that do the job for you (like WinZIP for example).

Change into the build directory of the decompressed package

```
cd resLoadSIM-x.y.z/build
```

and start the build process using CMake

```
cmake .
```

Don't forget the dot after `cmake`!
CMake checks whether everything needed for resLoadSIM is in place. If everything is ok, then CMake generates the makefiles needed for compiling the package. Start the compilation using make:

```
make
```

To install resLoadSIM type

```
make install
```

If no errors occur, then at this point ResLoadSIM is compiled, linked and ready to be used.

CMake will by default create a makefile for the optimized version of ResLoadSIM. If you need the debug version you have to call CMake with the following option:

```
cmake -DDEBUG:BOOL=ON .
```

For the parallel version of ResLoadSIM call

```
cmake -DPARALLEL:BOOL=ON .
```

1.3 Web Interface

Some additional steps are necessary to make use of ResLoadSIM's web interface. First of all you need to setup a HTTP server on your machine. Unless you have already a server running, get Apache: http://httpd.apache.org/. It comes with detailed documentation, so I won't cover that here. In the following I assume that the web server is up and running. You can check that under Linux or Mac OS in a terminal window by using the *ps* command for example. There should be a process named *httpd* somewhere in the output of *ps*:

```
ps -ef | grep httpd
```

You can also check the web server by typing the IP address of your machine in the address bar of your web browser. If the server is configured correctly the browser should show at least some kind of test page.

The next step is to make the web server present the ResLoadSIM web interface. This is accomplished by properly setting the variable *DocumentRoot* in the web server's configuration file. The name of the configuration file is usually *httpd.conf* - at least in case you use Apache. The file resides in */etc/apache2/* (Linux) or */private/etc/apache2* (Mac OS X). The web interface of ResLoadSIM is stored in resLoadSIM-x.y.z/www, which means that the DocumentRoot of your *httpd.conf* should point exactly there. Then restart the server to make it aware of the change:

```
apachectl restart
```

Now use your browser, type the address of your own machine and you should see the web interface of ResLoadSIM.

The next step is to setup the CGI scripts such that the user input on the interface can be processed properly. The source files of the 2 scripts are named *download.c* and *simulate.c* and are both located in the *www* subdirectory of the ResLoadSIM package. You can compile them manually or simply call

```
make cgi
```

from within the build directory. This will produce the scripts *download.cgi* and *simulate.cgi*. Then you have to move these scripts to a place where the web server can find them. On Mac OS X this would be */Library/WebServer/CGI-Executables/*. You can find the proper place by looking at the web server configuration file. The respective variable is called ScriptAlias. Once the scripts are in place everything should work just fine.

2 Execution

2.1 Command-line Interface

Most of the parameters are read from configuration files, but two of them have to be passed to ResLoadSIM on the command-line. These are the number of households and the number of days. A simulation of 10.000 households over a period of three days would be initiated as follows:

```
resLoadSIM 10000 3
```

The parallel version is initiated by using the *mpirun* command, which is part of the MPI installation, e.g.:

```
mpirun -np 4 resLoadSIM 10000 3
```

In the above eample resLoadSIM would be executed on 4 cpus (or cores). Using the -v option prints the current version (number, debug/optimized/parallel) of ResLoadSIM:

```
resLoadSIM -v
```

If you dont‘t want to see any output on screen while simulating then run resLoadSIM in silent mode by using the -s option:

```
resLoadSIM -s 1000 365
```

2.2 Using the Web Interface

The web interface is nothing more than a way to change the settings in the main configuration file *resLoadSIM.cfg* by toggling radio buttons and check boxes. The two required arguments, households and days, are entered in the text edit fields at the top. Once the 'Simulate' button is pressed, the script *simulate.cgi* reads the values from all entry fields and buttons, creates *resLoadSIM.cfg* on the fly, and starts the simulation. The simulation results are presented in a new browser tab, where the user has the possibility to download the results by pressing the 'Download Results' button.

2.3 Configuration Files

There are several configuration files. Location dependent information like coordinates, UTC offsets, ambient temperatures and solar irradiation data are stored in the locations directory. There is a subdirectory for each location, which contains a file named location.json and a temperature/irradiation data file from PVGIS, a tool located at http://re.jrc.ec.europa.eu/pvgis/

The second configuration file, *resLoadSIM.cfg*, contains parameters which control the behavior of the simulation. It is basically a list of keyword-value pairs, so called settings, interspersed with comments (lines starting with a hashmark). Some of the settings are organized in groups, enclosed by curly braces. The settings can be given in any order, and not all of them need to be specified. It is actually possible to run ResLoadSIM without *resLoadSIM.cfg* at all. Missing settings are substituted by default values. Information about what values have been used is written to a log file named *resLoadSIM.log*. The format of this log file is the same as the format of the configuration file. Therefore it is possible to use the log file as the configuration for a subsequent simulation just by renaming it to *resLoadSIM.cfg*. This is especially handy when you start from scratch without your own configuration. Just run a short dummy simulation, edit the settings in the resulting logfile (all defaults) according to your needs, and rename the file to *resLoadSIM.cfg*.

The following settings are known to ResLoadSIM:

location

String value
Default: Ulm
The following locations are available:
Amsterdam, Berlin, Madrid, Oslo, Paris, Rome, Ulm and Vienna
You can add a location by adding a subdirectory with name of the new location under the locations directory. This new subdirectory must contain a file named *location.json* and a PVGIS data file that contains the temperature and irradiation data of the new location.

control

Integer value in the range [0, 4]
Default: 0
This parameter sets the control mode of the producer.
0: No control. The simulation of the producer is simply bypassed.
1: Peak shaving. The producer tries to keep the load below a given limit (see peak_shaving_threshold).
2: The producer tries to control the load such that it matches a predefined load profile. The load profile is read by ResLoadSIM from a file named *profile*.
3: The producer tries to compensate a gap between projected and actual production.
4: Decentralized control via the electricity tariff.

peak_shaving
{

relative

Boolean value
Default: TRUE

threshold

Decimal value
Default: 85.0
This value is used only if the setting **control** = 1
When **relative** is TRUE, **threshold** is interpreted as a percentage (a value between 0 and 100) of the maximum load. When **relative** is FALSE, **threshold** is given in kWh.

}

seed

Integer value
Default: 0
Seed of the random number generator. If set to 0, the current time is used as seed. Any value greater than 0 is interpreted as seed itself.

output

Integer value in the range [0, 2]
Default: 1

0:	All output data is written to a single file.
1:	One output file per appliance.
2:	Both. One file per appliance plus one file with all data.

powerflow
{

step_size
Integer value >= 0
Default: 0
If set to 0, the powerflow solver is not used. Any value greater than 0 is interpreted as the number of timesteps between two invocations of the powerflow solver.

case_file_name
String value
Default: empty string
Name of the case file used by the powerflow solver. The format is explained at the end of section 2.3.

uv_control
Boolean value
Default: FALSE
Turn on/off undervoltage control

uv_lower_limit
Decimal value
Default: 0.910
Lower limit for undervoltage control

uv_upper_limit
Decimal value
Default: 0.925
Upper limit for undervoltage control

ov_control
Boolean value
Default: FALSE
Turn on/off overvoltage control

ov_lower_limit
Decimal value
Default: 1.075
Lower limit for overvoltage control

ov_upper_limit
Decimal value

Default: 1.090
Upper limit for overvoltage control

output_level

Integer value
Default: 1
0: no output related to the power flow solver
1: transformer files only
2: transformer files, partial input/output of the power flow solver
3: transformer files, full input/output of the power flow solver

}

start

{

These settings define the date and time for the simulation start.

day

Integer value in the range [1, 31]
Default: 1

month

Integer value in the range [1, 12]
Default: 4

year

Integer value
Default: 2015

time

Decimal value in the range [0.0, 24.0]
Default: 0.0

}

transient_time

Decimal value
Default: 1.0
The number of days of the pre-simulation run. In some cases it is desirable to add a few simulation days so that potential transient oscillations have time to settle before the actual simulation starts. In other cases this is even a necessity:

• When using peak-shaving we need to have a reference value for the maximum power peak
• In case there are households with a PV installation with batteries we need to initialize the batteries properly.
• Some appliances like the TVs need one day for initialization
• If the annual production of the PV modules is given as a fraction of the annual household consumption, the pre-simulation run has to be extended by a whole year to get values for the annual consumption.

daylight_saving_time

Integer value in the range [0, 2]
Default: 1
0: no daylight saving time (wintertime only)
1: standard daylight saving time
2: permanent daylight saving time (summertime only)

timestep_size

Decimal value
Default: 60.0
The timestep size given in seconds.

simulate_heating

Boolean value
Default: TRUE
Activate the simulation of the space heating demand?

price_grid
{

profiles

A list of price profiles. Each profile is enclosed in parentheses and contains a set of arrays. Each array consists of 3 floating point values: start time, end time (in hours) and the price per kWh. A profile covers a whole day.
Default: ([0.00, 6.00, 0.20][6.00, 22.00, 1.00][22.00, 24.00, 0.20])

sequence

An array of integer values.
Default: [1]
The values in this array are indices pointing to members of the list of profiles. The sequence [1, 3, 2] for example means that the simulation will take the first profile for day 1, the third profile for day 2 and the second profile for day 3. After that the sequence will be repeated, so day 4 uses the first profile, and so on.

}

price_solar
{

profiles

For a description refer to **price_grid** above
Default: ([0.0, 24.0, 0.10])

sequence

For a description refer to **price_grid** above
Default: [1]

}

Household related parameters are stored in *households.cfg*, and all appliance related parameters can be found in *tech.cfg*. Both files are country dependent and reside in one of the subdirectories of *countries/*. For a detailled description of all parameters in *households.cfg* and *tech.cfg* refer to sections 3.2 and 3.3.

If the parameter **powerflow.case_file_name** is not the empty string, then ResLoadSIM will use the file with the given name as the powerflow solver‘s case data file. A description of the Matpower case file format can be found here:
http://www.pserc.cornell.edu/matpower/docs/ref/matpower5.0/caseformat.html

ResLoadSIM specific extensions to the case data can be defined in a separate file which has to have the same name as the case data file with the appended suffix '.ext'. This file contains the relation between bus numbers and household numbers. It is possible to assign several households to a single bus. Each line in the file contains the following information:

The bus number
A boolean flag ('f', 'F', 't' or 'T') to specify whether it's a bus with solar households
The number of attached households
The list of household numbers

2.4 Output Files

The simulation results are stored in a set of output files, some of which can be used as input for a data plotting program like Gnuplot. The files are:

appliances.[year].[residents]

One file per year and per household size is generated. The output is triggered every year at December 31st and once when the simulation ends. Each file contains the yearly consumption of all appliances for the given year and size of households. There is one line per household, and the consumption values within a line are grouped by appliance type in alphabetical order and within each appliance type split with regard to the energy classes. That means the first values in each line refer to the air conditioners, followed by boilers, circulation pumps and so on. The number of values for each appliance corresponds to the number of energy classes of this appliance. For appliances, which have no energy classes (or for which energy classes are not implemented yet), only one value is print to the file.

battery.[year]

This one is produced only in case there are households with a photovoltaic installation including a battery. The 6 data columns are:

- the time in hours
- the mean charge of all batteries in percent
- the current power used for charging in kWh
- the current power drawn from all batteries in kWh
- the power loss while charging in kWh
- the power loss while discharging in kWh

This file is generated once per year (December 31st) and at the very end of the simulation.

consumption.[year]

This is a table that shows the total consumption of electricity in kWh during the year quoted in the file name. This is not necessarily a whole year, depending on the start and end date of the simulation. Each column stands for one household category (1 person, 2 persons, ...). The rows show the minimum, average and maximum consumption together with the median and the standard deviation for all household appliances. For each appliance the number of households with such an appliance and the total number of appliances of this type are given.

costs.[year]

There are two tables stored in this output file, both showing the mean consumption and the mean costs for all household categories. The two tables refer to households with and without photovoltaic installations.

dist.[year].[residents]

For each year and household category one distribution file is produced. The distribution is generated in the following way:
The interval between the minimum consumption and the maximum consumption of the households for the given category is divided in subintervals with constant length. Each household determines to which subinterval its own consumption belongs to. The middle point of each subinterval together with the number of households that belong to that interval are written to the file.

gridbalance.[year]

The five columns are:
the time (hours)
power into grid minus power from grid (kWh)
power from grid (kWh)
power into grid (kWh)
power from grid, which is used for charging the batteries (kWh).

households.[year].[residents]

One line per household is printed. Each line consists of the following values:
Household id
Number of residents
Total consumption in kWh
Own consumption of electricity produced by solar module in kWh
Energy drawn from battery in kWh
Nominal power of the solar module in kW (0 if not installed)
Production of the solarmodule in kWh
Battery capacity in kWh
Efficiency of charging
Efficiency of discharging
Costs for energy drawn from the grid
Income for energy fed into the grid

max.[year].[residents]

One line per household without a solar module. Each household stores the 3 highest power values (in kW) together with the times when they occured (in hours). One line consists of the following values:
Household id
Time 1
Power value (highest)
Time 2

Power value (2nd highest)
Time 3
Power value (3rd highest)

max_sol.[year].[residents]

One line per household. Only households with a photovoltaic installation are considered. Times are in hours and power in kW. The lines consist of:
Household id
Time 1
Household power at time 1 (highest)
Solar power at time 1
Time 2
Household power at time 2 (2nd highest)
Solar power at time 2
Time 3
Household power at time 3 (3rd highest)
Solar power at time 3
Time 4
Max. power drawn from grid at time 4 (highest)
Household power at time 4
Solar power at time 4
Time 5
Max. power drawn from grid at time 5 (2nd highest)
Household power at time 5
Solar power at time 5
Time 6
Max. power drawn from grid at time6 (3rd highest)
Household power at time 6
Solar power at time 6

power.[year].[appliance]

For each year and appliance there is one power file written to the disk. On top of that there is one power file for the households, which is named *power.[year].Total*.
The first column in each file is the time in hours. The second column is the power of all appliances of a certain type (e.g. all TVs) in all households, which are turned on at that time. In the next six columns the power is broken down in the six household categories. The file *power.[year].Solarmodule* contains an additional column showing the produced power that is used up by the household immediately. All power values are in kW, the time is given in hours.

summary.[year]

There are several sections. The first one shows the total consumption of all electrical appliances plus batteries during the year quoted in the file name. The second part refers to the photovoltaic installations. It shows the total solar electricity production, the part that is fed into the grid and the part that is used immediately. The third section shows the amount of heat generated by the solar collectors and how much is used for space heating and domestig hot water respectively. The following two sections show the amount of energy used for space heating and domestic hot water broken down by the type of heat source used.

If the powerflow simulation is used and the parameter *powerflow.output_level* is not set to 0, then additional output is generated:

trafo.[nr]

There is one file for each transformer in the grid. The values in the columns are:
the time (hours)
number of the bus with minimum voltage
minimum voltage magnitude
fraction of households in energy conservation mode
number of the bus with maximum voltage
maximum voltage magnitude
fraction of households in energy burst mode
power transmitted by transformer
maximum power
total power drawn by all households attached to this transformer
total solar production of all households attached to this transformer
household with maximum consumption
bus with maximum consumption

pfin/

A directory that contains the case data files, which are generated by resLoadSIM and used by the powerflow solver as input.

pfout/

This directory contains the output generated by the powerflow solver.

3 Programmer's Manual

3.1 Directory Structure

resLoadSIM-x.y.z/	Toplevel directory with the version number x.y.z
AUTHORS	Names of the program authors
CHANGELOG	The project's history as a list of changes
INSTALL	Very short installation instructions
build/	
CmakeLists.txt	Makefile used by the CMake build system
countries/	Country specific configuration files
doc/	
Manual.pdf	This document
include/	Contains the header files
locations/	Location related information
resLoadSIM.cfg	Main configuration file. Stores simulation parameters
src/	Contains the source files
www/	Contains all files related to the web interface
download.c	CGI script for downloading simulation results
error.html	Page shown whenever an error (invalid user input) occurs
images/	Images shown on the web interface
index.html	Main page of the interface
plot	Gnuplot batch file
simulate.c	CGI script for starting a simulation

3.2 Households

The parameters, which influence the behaviour of the households, are stored in the file *households.cfg* inside the *countries* directory. They are:

size_distribution[]
This vector stores the distribution of households of different sizes (in %). E.g. *size_distribution[3] = 12* means, that 12% of all households have 4 residents (vector index starts at 0).

retired_1
Percentage of single-person households where the resident is retired.

retired_2
Percentage of two person households where the residents are retired.

min_area[]
max_area[]
Minimum and maximum household area depending on the number of residents.

temperature_set
Set room temperature in °C
This value is used for calculating the space heating demand.

min_init_laundry
max_init_laundry
Minimu and maximum amount of laundry (in kg) at simulation start.

min_delta_laundry[]
max_delta_laundry[]
The minimum and maximum amount of laundry, which gets added to the current amount of laundry every day. The actual value is a random number between the two. Both parameters are vectors, because these values depend on the number of residents.

second_fridge[]
Probability of having a second fridge (in %), depending on the number of residents.

second_tv[]
Probability of having a second TV (in %), depending on the number of residents.

third_tv[]
Probability of having a third TV (in %), depending on the number of residents.

second_computer[]
Probability of having a second computer (in %), depending on the number of residents.

min_vacuum_interval
max_vacuum_interval
Minimum and maximum interval of vacuum cleaning in days.

light_factor[]
Used to calculate the number of lamps in a household. Depends on the number of residents.

rnd_wakeup[4]
rnd_wakeup_weekend[4]
rnd_wakeup_retired[4]
Used to calculate the wakeup time of a household. The four parameters of

each vector are the mean, sigma, lower and upper limit of a normal distributed random number. There are differnt vectors for different occasions (weekend, household with retired residents). All values are in seconds.

rnd_bedtime[2]
rnd_bedtime_weekend[2]
rnd_bedtime_retired[2]
Used to calculate the bedtime of a household. The two parameters of each vector are mean and sigma of a normal distributed random number. There are different vectors for different circumstances. All values are in seconds.

at_home_param[7]
These parameters are used to calculate the elements of the *at_home* matrix inside the household class. This matrix stores information about how many of the residents are at home at a given daytime. The first column of the matrix contains the time in seconds, the second column represents the number of residents. Whenever the number of residents changes, because someone leaves or returns home, this is represented as one row in the matrix. There is also one entry for the end of a day. Consider a single-person household for example, where the resident goes to work at 8 a.m. and returns home at 5 p.m. This would be stored in the *at_home* matrix this way:

```
at_home[0][0] = 28800;        // until !! 8 a.m.
at_home[0][1] = 1;            // the resident is at home
at_home[1][0] = 61200;        // between 8 a.m. and 5 p.m.
at_home[1][1] = 0;            // he is not at home
at_home[2][0] = 86400;        // for the rest of the day
at_home[2][1] = 1;            // he is at home again
```

Parts of this matrix are initialized randomly. This is where the parameter array *at_home_params* comes into play:

at_home_param[0] is the interval in seconds between wakeup and the time someone leaves home.
at_home_param[1] and *at_home_param[2]* are the lower and upper limit of a randomly chosen interval of time in which one or more residents are away from home. *at_home_param[3]* is the probability that in a two person household both residents go to work. All parameters so far refer to households with less than three residents.
at_home_param[4], [5] and *[6]* are for households with more than two residents and have the same meaning as *at_home_param[0], [1]* and *[2]*.

energy_class[9]
This vector defines the percentages of households, which have the energy label A+, A, … , H.

rnd_heat_source[5]
A vector of probabilities used to determine what heat sources a household uses to satisfy its heat demand for space heating (SH) and domestic hot water. (DHW). The 5 currently implemented heat sources are:
oil, gas and district heating (for both SH and DHW), heat pumps with

electrical heating as backup, and finally solar collectors together with heat pumps as backup and a water storage as buffer.

prevalence
This is a group settings. It contains the prevalence arrays for all household appliances. For example, the prevalence array for freezers could look like this:
freezer = [17.00, 40.00, 50.00, 55.00, 60.00, 65.00];
That means that 17% of all single person households, 40% of all two person households, and so on, own a freezer.

3.3 Appliances

The core of the simulation are the appliances' *simulate* functions. They model the appliances' behaviour depending on several, partially random, factors. Each appliance can be in one of two states: ON or OFF. All the *simulate* function does, is to determine the points of time, when an appliance has to change its state. This is usually done in advance at the start of each day (daytime == 0). During the simulation the current daytime is then compared to those saved points of time. In some cases though a timer is used to keep track of an appliance‘s state. For example when a washing machine is turned on we know the cycle time and set a timer accordingly.

All appliances, which are currently turned on, contribute to the total power consumption of a household. This is taken care of at the very end of each *simulate* function. In the following a description of all application related configuration parameters is presented for all appliances. All parameters are defined in the country dependent technical configuration *tech.cfg*. The flow charts for the appliance *simulate* functions can be found in *doc/FlowCharts*.

3.3.1 Fridges

Fridges are controlled by the following parameters:

min_temperature
The minimum temperature of a fridge

max_temperature
The maximum temperature of a fridge

smartgrid_enabled
Percentage of smartgrid enabled fridges

energy_class[10]
This vector defines the percentages of fridges, which have the energy label A+++, A++ and so on.

smart
Percentage of smart fridges. These are fridges which try to make use of

photovoltaic overproduction.

delta_t_rise_factor
delta_t_rise_mean
delta_t_rise_sigma
Parameters used to calculate how much the temperature of a fridge rises within one timestep according to this formula: *factor * N(mean, sigma)*
N(mean, sigma) is a normal distributed random number.

delta_t_drop_factor
delta_t_drop_mean
delta_t_drop_sigma
These parameters are used to calculate how much the temperature of a fridge drops within one timestep.

Vc_mean[]
Vc_sigma[]
Vc_low[]
Vc_high[]
Parameters to define the size of the fridge compartment. The size is a normal distributed random number with a lower and an upper limit. All parameters are given as vectors, because they depend on the number of residents per household.

Tc
This is the nominal temperature of the fridge compartment, which is used in the formula to calculate the equivalent fridge compartment volume, which in turn is used to calculate the standard annual energy consumption. The latter is part of the formula to calculate the fridge‘s power consumption. All that formulas can be found in annex XIII of this document:
http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32010R1060&from=EN

power_factor
The ratio of real power to apparent power.

3.3.2 Freezers

The parameters for freezers are the same as those for fridges in the previous section, with the following exceptions:

Vc_per_resident
Freezer compartment size per resident.

mn_power_factor
Part of the formula for the standard annual energy consumption are the parameters M and N, which come in two versions (M=0.539, N=315 and M=0.472, N=286). **mn_power_factor** is interpreted as a percentage. For example: if it is set to 70, it means that for 70% of all freezers the first set of numbers for M and N is used.

The formula for the standard annual energy consumption can be found in annex XIII of this document:
http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32010R1060&from=EN

power_factor
The ratio of real power to apparent power.

3.3.3 Boilers

energy_class[10]
This vector defines the percentages of boilers, which have the energy label A+++, A++ and so on.
The sum of all values in this array has to be 100 obviously.

power_factor
The ratio of real power to apparent power.

3.3.4 Air conditioners

The air conditioners are still under construction.

power
The power (in kW) for ALL air conditioners in the simulation.

power_factor
The ratio of real power to apparent power.

The current simulation model is nothing more than a placeholder for a real model, which has to be implemented yet. For the time being an air conditioner is turned on at the household's wakeup time and is turned off at bedtime.

3.3.5 Dishwashers

Parameters for dishwashers:

smartgrid_enabled
Percentage of smartgrid enabled fridges

energy_class[7]
This vector defines the percentages of boilers, which have the energy label A+++, A++ and so on.

smart
Percentage of smart dishwashers. These are appliances, which try to make use of photovoltaic overproduction.

hours_per_cycle
Time in hours for one cycle

capacity[2]
The dishwasher‘s capacity is assumed to grow linearly with the number of residents:
*capacity = capacity[0] + capacity[1] * residents*

SAEc_small[2]
Two parameters used to calculate the standard annual energy consumption for small dishwashers.

SAEc_big[2]
Two parameters used to calculate the standard annual energy consumption for big dishwashers.

factor
Multiplier used in the formula for the dishwasher‘s power.

probability[2]
Two parameters used to calculate the chances that the dishwasher is used at a given day.

ignore_price
In a simulation with price control enabled this number stands for the probability (in percent) that a dishwasher will ignore the current price for electricity.

fraction
It is assumed that most dishwashers are used mainly shortly after lunchtime, or later in the evening. The fraction parameter is interpreted as the percentage of dishwashers, which are used rather earlier than later.

timer_1_mean
timer_1_sigma
The parameters of a normal distributed random number which is used as a timer. This timer determines when a dishwasher is turned on. Timer_1 is used for dishwashers that start rather early.

timer_2_mean
timer_2_sigma
The parameters of a normal distributed random number which is used as a timer. This timer determines when a dishwasher is turned on. Timer_2 is used for dishwashers that start later in the day.

timer_3_mean
timer_3_sigma
Parameters of a normal distributed random number used as a timer for smart dishwashers, which are turned on after sunset. It is used to set the amount of time (in seconds) between sunset and the start of the dishwashers.

preview_length
This is the number of minutes a dishwasher under price control will look into the future to find the lowest price for electricity.

peak_delay
If peak shaving is enabled and the electricity provider has issued a stop signal, then smartgrid enabled dishwashers will delay their start for a number of seconds, which is defined by this parameter.

power_factor
The ratio of real power to apparent power.

3.3.6 E-Vehicles

smartgrid_enabled
Percentage of smartgrid enabled e-vehicles.

smart
Percentage of e-vehicles, which try to charge their batteries when there is a surplus production of solar electricity.

min_battery_capacity
max_battery_capacity
The minimum and maximum battery capacity, which is assumed to be uniformly distributed between these two parameters.

power
The power drawn when charging the battery. Currently only one value for all vehicles.

departure_delay
Time interval in seconds between household wakeup time and departure time of the vehicle.

arrival_time_mean
arrival_time_sigma
The arrival time of the e-vehicle is a normal distributed random number with *mean* and *sigma* as parameters. The numbers are in seconds.

delta_time
Smartgrid enabled e-vehicles under price control look *delta_time* seconds into the future to find the lowest electricity price.

lower_limit
upper_limit
When the e-vehicle is used, the battery charge drops a certain amount, which is assumed to be a random number uniformly distributed between *lower_limit* and *upper limit*. Both parameters are given as a fraction of the total battery capacity.

fraction_used
Not all cars will be used every day. This parameter is the percentage of cars that actually hit the road.

ignore_price
In a simulation with price control enabled this number stands for the probability (in percent) that an e-vehicle will ignore the current price for electricity.

3.3.7 Circulation Pumps

controlled
Percentage of controlled circulation pumps, which are turned on and off during daytime, as opposed to uncontrolled pumps, which just run permanently between morning and evening.

power_per_size
Given in kW. Used to calculate the pump‘s power:
*power = power_per_size * household_area*

rnd_time_on[2]
The length of the time interval a controlled pump is turned ON is calculated as a random number, which is uniformly distributed between *rnd_time_on[0]* and *rnd_time_on[1]*. The numbers are in seconds.

rnd_time_off[2]
The length of the time interval a controlled pump is turned OFF is calculated as a random number, which is uniformly distributed between *rnd_time_off[0]* and *rnd_time_off[1]*. The numbers are in seconds.

rnd_first_day[4]
The first day a pump gets turned on is calculated as a normal distributed random number *N(rnd_first_day[0], rnd_first_day[1])*. *rnd_first_day[2]* and *rnd_first_day[3]* are the lower and upper limit.

rnd_last_day[4]
Analogous to *rnd_first_day[4],* these parameters are used to calculate the day when a pump gets turned off again.

first_month
The first month in a year a pump gets turned on.

last_month
The month when the pump gets deactivated.

time_1
Daytime in seconds when a non-controlled pump is turned OFF

time_2
Daytime in seconds when a non-controlled pump is turned ON

power_factor
The ratio of real power to apparent power.

3.3.8 Computers

power
Power drawn by a computer in kW.

power_factor
The ratio of real power to apparent power.

duration_mean
duration_sigma
The total time a computer is turned on during the day is calculated as a normal distributed random number with these two parameters. Both are in seconds.

duration_fraction
The total runtime is split into two intervals. The length of the first interval is given as a fraction of the total runtime, therefore *duration_fraction* is a value between 0 and 1.

duration_fraction_saturday
This is the *duration_fraction* used on Saturdays.

duration_fraction_sunday
This is the *duration_fraction* used on Sundays.

time_offset[3]
time_offset_saturday[3]
time_offset_sunday[3]
Interval in seconds between the household wakeup time and the first time the computer gets booted.
These values come in sets of three to make things a bit more interesting. Which one of the values is used is determined by the random number generator.

rnd[3]
rnd_saturday[3]
rnd_sunday[3]
These are the values used by the random number generator to select the previously described *time_offset*. All values are read as percentages.

time_2_mean
time_2_sigma
Daytime in seconds when the second runtime interval of a computer starts. Calculated as a normal distributed random number as usual.

3.3.9 Cookers

power[]
The power of a cooker dependent on the number of residents in the households.

power_factor
The ratio of real power to apparent power.

duration_1_percent
The total time per day a cooker is in use is split in parts. *duration_1_percent* is the probabilty that the cooker is used in the morning.

time_offset
This is the time in second between the household wakeup time and the time the cooker gets activated.

duration_2_percent
duration_2_percent_saturday
duration_2_percent_sunday
This is the probabilty that the cooker is used for preparing lunch. The values can differ depending whether it is a weekday, a Saturday or a Sunday.

rnd_duration_1[4]
rnd_duration_2[4]
rnd_duration_3[4]
The duration a cooker is turned on is calculated as a normal distributed random number. Each array contains four values: mean, sigma, lower and upper limit of this random number. *rnd_duration_1* is used in the morning, *rnd_duration_2* at lunchtime, and *rnd_duration_3* in the evening.

time_2_mean
time_2_sigma
Parameters (in seconds) for a normal distributed random number which is the daytime when the cooker is turned on for preparing lunch.

time_3_mean
time_3_sigma
Parameters (in seconds) for a normal distributed random number which is the daytime when the cooker is turned on for preparing dinner.

3.3.10 E-Heating

smartgrid_enabled
Percentage of smartgrid enabled electrical heatings

power_factor
The ratio of real power to apparent power.

kW_per_m2
Power per square meter household area in kW.

3.3.11 Lights

energy_class[7]
This vector defines the percentages of lights, which have the energy label A+++, A++ and so on.

power_factor
The ratio of real power to apparent power.

sigma_morning
sigma_evening
Used to calculate for how long lights get turned on in the morning and in the evening.

luminous_flux_mean
luminous_flux_sigma
luminous_flux_min
luminous_flux_max
The luminous flux, which is used to calculate the power consumption, is calculated as a normal distributed random number with mean, sigma, lower and upper limit as parameters.

3.3.12 Tumble Dryers

smartgrid_enabled
Percentage of smartgrid enabled tumble dryers.

energy_class[7]
This vector defines the percentages of tumble dryers, which have the energy label A+++, A++ and so on.

power_factor
The ratio of real power to apparent power.

min_capacity
Minimum capacity in kg.

max_capacity
Maximum capacity in kg.

ignore_price
Percentage of dryers which ignore the price of electricity when under price control.

peak_delay
If peak shaving is enabled and the electricity provider has issued a stop signal, then smartgrid enabled dryers will delay their start for a number of seconds,

which is defined by this parameter.

3.3.13 TVs

energy_class[10]
This vector defines the percentages of TVs, which have the energy label A++ +, A++ and so on.

power_factor
The ratio of real power to apparent power.

diagonal_1
diagonal_2
diagonal_3
Size of a TV in inch for the first, second and third TV in a household.

avg_duration[]
Average time (seconds) a TV is turned on per day, depending on the number of residents in a household.
avg_duration[0] → single-person household, *avg_duration[1]* → two person household, ...

factor_mean
factor_sigma
The total time a TV is turned on per day is a normal distributed random number *N(factor_mean*avg_duration, factor_sigma*avg_duration)*.

factor_mean_we
factor_sigma_we
Same as above, but used for Saturday/Sunday.

duration_factor
duration_factor_sat
duration_factor_sun
Fraction of the total time a TV is on (weekday, Saturday and Sunday).

random[3]
random_sat[3]
random_sun[3]
These are the values used by the random number generator to determine when to turn on the TV for the first time during the day. All values are read as percentages.

delay[3]
delay_sat[3]
delay_sun[3]
When TVs are turned on in the morning, this happens *delay* seconds after the household wakeup time. Values differ depending on whether it is a weekday or Saturday/Sunday.

time_2_mean
time_2_sigma
Parameters for a normal distributed random number, which is interpreted as the daytime when a TV is turned on for the second time. Values are in seconds.

3.3.14 Vacuum Cleaners

energy_class[10]
This vector defines the percentages of vacuum cleaners, which have the energy label A+++, A++ and so on.

power_factor
The ratio of real power to apparent power.

timer_min
timer_max
The daytime (in seconds) when to use the vacuum cleaner is calculated as a random number between *timer_min* and *timer_max*.

timer_sigma
timer_factor
Both parameters are used to calculate for how long to turn on the vacuum cleaner (in seconds).

3.3.15 Washing Machines

energy_class[10]
This vector defines the percentages of washing machines, which have the energy label A+++, A++ and so on.

power_factor
The ratio of real power to apparent power.

smartgrid_enabled
Percentage of smartgrid enabled washing machines.

smart
Percentage of washing machines, which try to turn themselves on when there is a surplus production of solar electricity.

hours_per_cycle
Time in hours for one washing cycle.

capacity
How much laundry fits into the washing machine (in kg).

random_limit
This value is interpreted as a percentage. Together with other factors this parameter is used to determine whether the washing machine gets fired up at

a given day.

ignore_price
In a simulation with price control enabled this number stands for the probability (in percent) that a washing machine will ignore the current price for electricity.

best_price_lookahead
The number of minutes a washing machine will look into the future to find the lowest price for electricity.

timer_mean
timer_sigma
Both parameters are in seconds and are used to set a timer for smart washing machines that had no chance to make use of a surplus production of the solarmodules. Those machines will be turned on a randomly chosen time after sunset.

peak_delay
If peak shaving is enabled and the electricity provider has issued a stop signal, then smartgrid enabled washing machines will delay their start for a number of seconds, which is defined by this parameter.

3.3.16 Heat Pumps

kW_per_m2
Power per square meter household area in kW.

power_factor
The ratio of real power to apparent power.

min_eff
max_eff
The efficiency of a heat pump is a random number between *min_eff* and *max_eff*. Both are values between 0 and 1.

min_temperature
max_temperature
The hot temperature of a heat pump is randomly choosen within min and max temperature.

3.4 Solarmodules and Batteries

The simulation of solar modules and attached batteries differs considerably from the simulation of the household appliances. The power of an appliance drawn from the grid or the battery is assumed to be constant over time, while this appliance is turned on, and can be calculated before the simulation starts. The power delivered from a solar module on the other hand varies over time and has to be updated every timestep. It can safely be assumed that the power before sunrise and after sunset equals zero. Between sunrise and sunset the power output of a solar module is calculated using its nominal power and the solar irradiation values at the given location. The latter values can be obtained by using PVGIS, a program which is found here:

http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php

Once the power output of the module has been calculated, it is compared to the household's current consumption. Any surplus production is fed into the grid or used to charge the battery, if available. In case the consumption is higher than the production, power is drawn from the battery. The simulation takes into account that there are losses while charging or discharging a battery, and that the battery's capacity, the charging and discharging power are limited.

Parameters related to the solarmodules are:

system_loss
Estimated system losses of the PV installation (in %)

production_ratio
The production ratio is the ratio of solar production to household consumption. If this value is zero, resLoadSIM will set the nominal power of all solar modules to random values depending only on the household size, and then perform an ordinary simulation. For any value bigger than zero the nominal power of the solar modules can be calculated precisely as long as the household consumption is known. Therefore, when production_ratio > 0, resLoadSIM will extend the transient time by 365 days.

min_area
max_area
The size of a solarmodule per household resident is a random number between *min_area* and *max_area* (in m²).

min_eff
max_eff
The efficiency of a solarmodule is a random number between *min_eff* and *max_eff*. Both are values between 0 and 1.

Parameters for the batteries are:

frequency_solar
The frequency (in %) of batteries in households with a photovoltaic installation.

frequency_non_solar
The frequency (in %) of batteries in households without a photovoltaic installation.

capacity_in_days
If this parameter is bigger than zero, it is interpreted as the number of days a fully charged battery can provide energy to a household without the need of a recharge. This requires to know a household‘s average consumption per day, which in turn makes it necessary to extend the simulation‘s transient time by 365 days (which is done automatically!!). If this parameter is set to zero, then the following two parameters are used to calculate the battery‘s capacity.

min_capacity_per_resident
max_capacity_per_resident
The capacity is set to a random value between *min_capacity_per_resident* and *max_capacity_per_resident* multiplied by the number of residents.

smartgrid_enabled
Percentage of smartgrid enabled batteries.

installation_costs
Fixed installation costs for a solar battery (in Euro)

avg_lifetime
The average lifetime of a battery in years.

min_price
max_price
The minimum and maximum retail price per kWh capacity of a battery in Euro.

min_eff_charging
max_eff_charging
The minimum and maximum efficiency while charging.

min_eff_discharging
max_eff_discharging
The minimum and maximum efficiency while discharging.

max_power_charging
The maximum power used for charging, measured as a fraction of the maximum capacity.

max_power_discharging
The maximum power while discharging, measured as a fraction of the maximum capacity.

allow_grid_charge_solar
Boolean flag to choose whether solar batteries are allowed to be charged with electricity from the grid.

3.5 Load Shifting

There are several modes of smart grid control implemented in ResLoadSIM.

They are user selectable via the setting *control* in the configuration file *resLaodSIM.cfg*. The control modes are:

None: This is the trivial case. Smartgrid control is disabled.

Peak Shaving: In this mode the simulation tries to keep the maximum consumption below a user specified threshold value (peak_shaving.threshold in *resLoadSIM.cfg*). This threshold refers to the maximum peak value, which has been calculated during the first day of the simulation. This has an effect on all smartgrid enabled appliances, which can be fridges, freezers, e-vehicles, washing machines, tumble dryers and dishwashers. The first three of those can be turned on and off by centralized control at any time, under the condition that the temperature of fridges and freezers does not exceed a user defined limit. For the rest of the smartgrid enabled appliances the producer sends a global stop signal whenever the total load reaches the peak shaving threshold. No washing machine, tumble dryer or dishwasher is turned on while this signal is in effect, but already running ones aren't interrupted.

Profile: The producer tries to control the total consumption such that it follows a user defined profile. It influences all smartgrid enabled appliances in the same way as described in the last paragraph. The profile is a file with 2 columns of values, which are the time in hours and the total consumption of all households at any given time.

Price: This is a decentralized way of influencing the load distribution by using the electricity tariff as an incentive for the households. Only the washing machines, the dishwashers and the tumble dryers are effected by this option.

3.6 Grid Voltage Control

By using the PETSc‘s power flow solver ResLoadSIM is able to simulate a reaction to grid voltage levels, which raise above or drop below predefined limits. The communication between ResLoadSIM and pflow is done via files. ResLoadSIM writes the grid topology together with the households‘ consumption values to a file in MATPOWER format. Pflow reads that file, calculates the voltage levels for each node in the grid and writes the results to a file, which is then read by ResLoadSIM.
If undervoltage control is enabled, ResLoadSIM will try to reduce the consumption of affected households by sending them a ‚reduce consumption‘ signal. When voltage levels rise back to normal, this signal is revoked. In order to avoid oscillations two mechanisms have been implemented. First, the signal is sent only to a fraction of all affected households (e.g. 10%) per timestep. Secondly, before the ‚reduce consumption‘ signal is revoked, the voltage has to rise above a limit, which is higher compared to the limit which triggered the signal in the first place (hysteresis).
Overvoltage control works analogously. In this case households, which are

affected by voltage levels above nominal voltage, are sent a ‚raise consumption‘ signal.

3.7 Space Heating

The simulation of the space heating demand is based on the hourly calculation procedure presented in the European Standard prEN ISO 52016-1, section 6.4. Since ResLoadSIM is used to simulate thousands of households of which we do not have detailled information regarding the layout, some simplifications had to be applied. E.g. it is assumed that all housholds are detached houses. There are no inner walls. Air circulation through open windows or doors are neglected as well as the heat input caused by solar irradiation through windows.

The households calculate their heat demand for space heating and propagate the result to the heat sources. Currently five different configurations of heat sources are supported:

1. Oil heating
2. Gas heating
3. District heating
4. Heat pumps together with electrical heating and boilers
5. Solar thermal collectors together with heat pumps

The first three configurations power space heating as well as water heating. The fourth combination of heat sources lets the heat pumps take care of space heating and uses electrical heating only as a last resort when the heat pumps cannot deliver enough heat power. The boilers are used for the preparation of hot water. The last configuration is unique in the sense that it uses a water storage as a buffer. The storage has been implemented in a similar way to the batteries used in connection with the solar modules. The model used for the solar collectors is based on the European Standard prEN 15316-4-3. The main input for the solar collector model is the solar irradiation data, which is provided by the PVGIS database: http://re.jrc.ec.europa.eu/pvgis/