PowerGAMA grid and market analysis

PowerGAMA
User guide (v1.1)

Harald G Svendsen

9th February 2018

Abstract

This document gives a rough and brief overview of the PowerGAMA software tool for optimal power flow analysis of large power systems, taking into account variability of demand and power generation from renewable sources.

Contents

1 Introduction

PowerGAMA is open source software created by SINTEF Energy Research. The expanded name is *Power Grid And Market Analysis*. This is a Python-based lightweight simulation tool for high level analyses of renewable energy integration in large power systems.

The simulation tool optimises the generation dispatch, i.e. the power output from all generators in the power system, based on marginal costs for each timestep for a given duration. It takes into account the variable power available for solar, hydro and wind power generators. It also takes into account the variability of demand. Moreover, it is flow-based meaning that the power flow in the AC grid is determined by physical power flow eqautions.

Since some generators may have an energy storage (hydro power with reservoir and consentrated solar power with thermal storage) the optimal solution in one timestep depends on the previous timestep, and the problem is therefore be solved sequentially. A realistic utilisation of energy storage is ensured through the use of storage values.

PowerGAMA does not include any power market subtleties (such as start-up costs, limited ramp rates, forecast errors, unit commitments) and as such will tend to overestimate the ability to accomodate large amounts of variable renewable energy. Essentially it assumes a perfect market based on nodal pricing without barriers between different countries. This is naturally a gross oversimplification of the real power system, but gives nontheless very useful information to guide the planning of grid developments and to assess broadly the impacts of new generation and new interconnections.

PowerGAMA is largely inspired by SINTEF's Power System Simulation Tool (PSST) [1, 2].

PowerGAMA source code and a wiki is found on this web address:
`https://bitbucket.org/harald_g_svendsen/powergama`.

1.1 Licence

PowerGAMA is open source software distributed under the following licence:

- The MIT License (MIT) (`http://opensource.org/licenses/MIT`)

1.2 Dependencies

PowerGAMA is written with Python, and is executed via a Python interpreter. It has the following dependencies, all of which are freely available:

- Python 3 or 2.7
- Python packages: numpy, scipy, matplotlib, mpl_toolkits, PuLP

1.3 Installation

The above dependencies must be installed before PowerGAMA can be executed. Convenient Python packages that includes the numpy, scipy and matplotlib packages and the Matlab-like development environment Spyder are *Anaconda* () and *Python(x,y)*. Installing additional python packages is generally straightforward, using e.g. *pip* or the "python setup.py install" command. This includes the installation of PowerGAMA itself.

2 Physical description

This section gives a brief overview of the physical description of the power system grid and market. This description includes a number of implicit assumptions and is the basis for the PowerGAMA software tool.

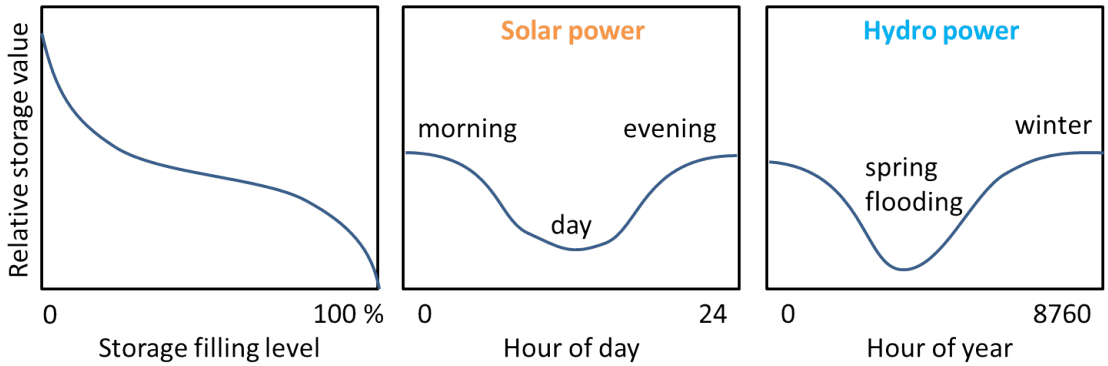

Figure 1: Illustration of typical storage values with dependence on filling level (left) and time variation for solar power with small storage (middle) and hydro with large storage (right)

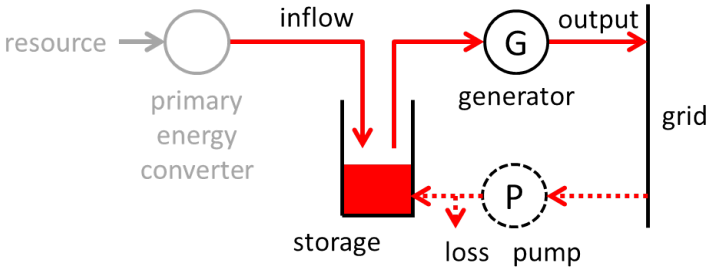

Figure 2: Power generation model model.

2.1 Electricity market

The market is considered perfect such that generators with the lowest marginal costs are always favoured. That is, power is assumed traded such that the overall cost of generation is always minimised.

The cost of generation is assumed to be the marginal cost times power output for all generators. Start-up costs and ramp rates are not considered.

For generators with a storage, the marginal cost is given by storage values, which depends on filling level. Storage values are inspired by watervalues used for modelling and planning of production for hydro power generators with storage [3, 4]. Storage values reflect the value of adding power inflow to the storage. The generator will therefore produce if the cost of alternative generation is higher than the storage value at any given time. This is identical to saying that a generator with storage will produce powr if the nodal price is higher than the storage value.

Storage values curves are given as input, as functions of storage filling level and time. If the storage is nearly full, the storage value is low, since adding to the storage may lead to energy spillage. If the storage is nearly empty, the storage value is high. For predictable seasonal or daily inflow patterns, the storage value is low just before a peak in the inflow, and high before a dip. An illustration of how the storage values may vary with filling level and time is given in Figure 1 for solar consentrated power with small storage (hours) and for hydro power with large reservoir for seasonal storage.

2.2 Power generation

Power generators are described by the same universal model, illustrated in Figure 2. Different types of power plants are simply distinguished by their different parameters, as indicated in Table 1. PowerGAMA assumes that the power inflow (average value and time profile) is given as input, and so the resource and primary energy converter parts included in Figure 2 are not directly relevant.

Wind and solar PV power are similar. The inflow represents the available electrical power in the wind or solar radiation. Zero storage implies that power not used is lost. The cost is zero, such that

Table 1: Generator parameters: Inflow, storage capacity and marginal cost for different types of generators

Generator type	Inflow	Storage	Price
Fuel based (alt 1)	0	Infinity	Constant cost
Fuel based (alt 2)	Available capacity	0	Constant cost
Wind	Available power	0	0
Photovoltaic	Available power	0	0
CSP	Available power	Thermal storage	Storage value
Hydro	Water inflow	Reservoir capacity	Storage value

unless restricted by grid constraints, available power will always be output. solar CSP and hydro power *without* any storage can also be modelled in this way.

CSP and hydro *with storage* will have a non-zero storage and a price set such as to ensure a sensible scheduling of the power output. If the storage is close to its upper limit, the price will be low and if it is low, the price will be high. This dependence on the storage filling level is exactly what the storage value provides. It gives the threshold price value for when the generator should produce power at a given storage filling level. The use of storage value is well established for scheduling of hydro power production. Storage values also depend on the time: The value of stored water is less just before the spring floods than in the autumn, and the value of stored CSP energy is less in the late morning (around 9am) than in the evening. However, this dependence will be ignored in this software.

For fuel based generators, such as coal, gas, oil, nuclear and biomass, there are two alternative ways to specify the inflow and storage parameters. *Alternative One* is to set inflow to zero and storage to infinity with an initial value also to infinity. Then there is always fuel available in the storage and output is restricted by generator output limits only. With this approach there is no spilled energy, but storage is updated each time step. *Alternative Two* is to set inflow equal to available capacity and storage to zero. This allows available capacity to vary throughout the year, which may be relevant at least for nuclear power and perhaps biomass. But it will wrongly indicate spilled energy whenever actual output is less than the available capacity.

For generators with a storage, a pump, or more generally, an energy storage charger, may be included in the model. Technically, a pump is represented as a generator with zero or negative output that takes energy from the grid and adds to the storage, with a certain amount of loss.

The generator model is described by the following parameters:

- Power inflow (P_{inflow})
- Energy storage capacity (E_{max})
- Energy in storage (E)
- Generator capacity (P_{max})
- Generator minimum production (P_{min}). Normally, this is value is zero.
- Generator output (P_{output})
- Generator available power (P_{avail})
- Pumping capacity ($P_{\text{pump,max}}$). In most cases, this value is zero (no pumping)
- Pumping demand (P_{pump})
- Pumping efficiency (η_{pump})

Available power is determined by the power inflow and the amount of stored energy, and is given by

$$P_{\text{avail}} = P_{\text{inflow}} + \frac{E}{\Delta t}, \tag{1}$$

where Δt is the time interval considered.

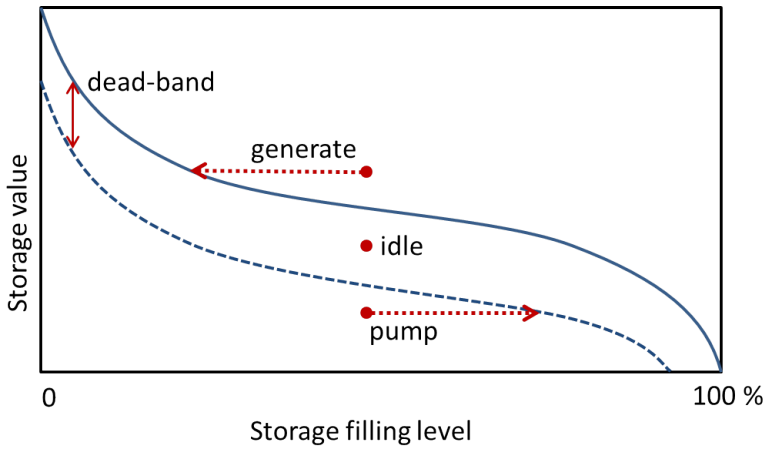

Figure 3: Generator with storage and pumping. The red dots represent three different situations with different nodal price at the generator associated node. The solid line is the storage value curve, and the dotted line is the same curve offset by a dead-band.

Generator output is limited both by the available power and by the generator capacity, as well as the minimum production. Regarding the upper limit, there is a difference depending on generator type. From the definition of power capacity for PV panels, it may occur in good conditions that the power output is higher than the nominal capacity. In this case output power should not be constrained by the nominal capacity, but only by the available power. In fact, this is true for all generators with zero storage. In general, therefore, the upper limit on generator output is given as

$$\text{No storage generators:} \qquad P_{\text{limit}} = P_{\text{avail}}, \tag{2}$$

$$\text{Non-zero storage generators:} \qquad P_{\text{limit}} = \min[P_{\text{avail}}, P_{\text{max}}], \tag{3}$$

$$\text{Infinite storage generators:} \qquad P_{\text{limit}} = P_{\text{max}}. \tag{4}$$

The constraint on generator power output can then be written

$$P_{\text{min}} \leq P_{\text{output}} \leq P_{\text{limit}}. \tag{5}$$

2.2.1 Power inflow

Power inflow is important for renewable energy sources such as wind, sun and rain. In general the energy resource may be converted by a primary energy converter (e.g. wind turbine) into a more readily exploitable energy form. The available power for electricity generation is generally less than the power in the primary resource. PowerGAMA assumes that power inflow data is provided by the user. For wind power, the inflow depends on the wind speed and characteristics of the wind turbines. Similarly, for solar power the inflow depends on solar radiation and characteristics of the PV or CSP units. For hydro power, the inflow depends on precipitation, temperature, topography and soil conditions.

2.2.2 Pumping

The implementation of pumping is illustrated in Figure 3. If the price is high, the generator will produce power, reducing the storage filling level. If the price is below the storage value the generator will be idle, allowing the storage to fill up (due to inflow). If the price is also lower than the storage value minus a certain dead-band value, the pump will add energy to the storage, increasing the filling level.

The dead-band ensures that the generator–pumping system doesn't continuously alternate between generating and pumping, and indirectly takes into account the losses associated with pumping.

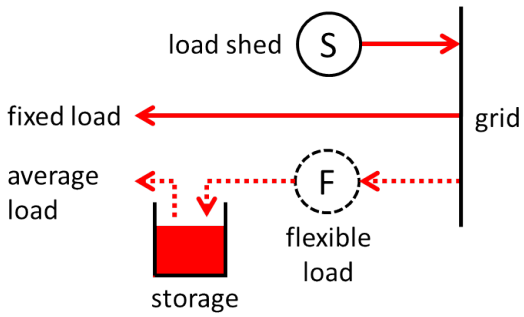

Figure 4: Power consumption model.

2.2.3 Storage dynamics

The energy in the storage E is given by the differential equation

$$\frac{dE}{dt} = P_{\text{inflow}} - P_{\text{output}} + \eta_{\text{pump}} P_{\text{pump}}, \quad 0 \leq E \leq E_{\max}, \tag{6}$$

where $E_{\max}$ is the maximum capacity of the storage. Discretised, this equation is written

$$E(t + \Delta t) = \max[\min[E(t) + \Delta t(P_{\text{inflow}} - P_{\text{output}} + P_{\text{pump}}), E_{\max})], 0] \tag{7}$$

2.3 Power consumption

Power consumption is modelled as loads connected to different nodes with a given average power demand and a certain profile which describes the time dependent variation, i.e. how the power consumptions varies from time step to time step. Any number of demand profiles can be specified and any number of loads can be connected to each node.

The full consumption model is illustrated in Figure 4, with futher explanaitions in the following.

2.3.1 Load shedding

Power demand that for some reason cannot be supplied is handled by adding high cost generators at all nodes with loads attached. This ensures that the optimisation has a solution. The high cost further ensures that these are used only when strictly necessary. Output from these "generators" is interpreted as load shedding.

2.3.2 Flexible load

Flexible loads are represented in a similar way to flexible generation, by means of a storage and storage value curves.

The main parameters specifying a flexible load are:

- The ratio of flexible energy to total energy demand ($P^{\text{avg}}_{\text{flex}}/P^{\text{avg}}$)
- The flexibility on/off ratio, i.e. ratio of average load relative to the maximum load ($x_{\text{flex}} = P^{\text{avg}}_{\text{flex}}/P^{\max}_{\text{flex}}$)
- The maximum energy flexibility/storage ($E^{\max}_{\text{flex}}$), given in MWh
- The storage value curve (normalised shape)
- The storage base value (absolute value), given in €/MWh

The amount of power served to the flexible load, P_{flex} is determined by the optimisation. Given this value, the energy in the storage E_{flex} for the flexible load is computed according to

$$E_{\text{flex}}(t + \Delta t) = E_{\text{flex}}(t) + \Delta t(P_{\text{flex}} - P^{\text{avg}}_{\text{flex}}), \tag{8}$$

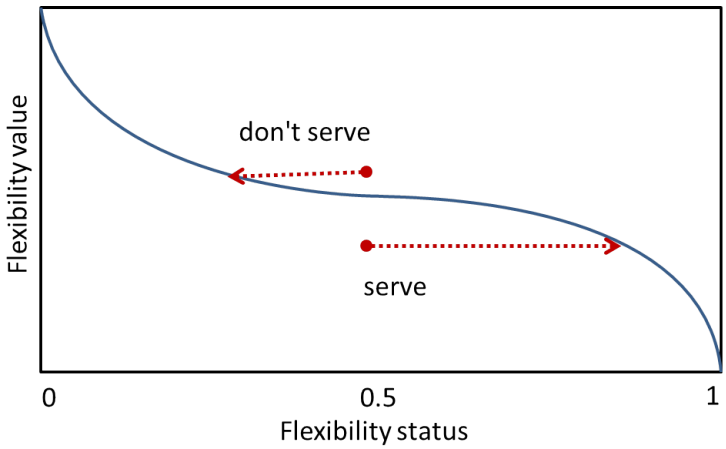

Figure 5: Storage value curve for flexible load. The red dots represent nodal prices in two different operating points

where Δt is one time step. The storage filling level f_{flex} is the relative value of energy in the storage compared to the maximum flexible energy (storage capacity),

$$f_{\text{flex}} = \frac{E_{\text{flex}}}{E_{\text{flex}}^{\max}}. \tag{9}$$

If $f_{\text{flex}} > 0.5$ then the load is "over-served", and if $f_{\text{flex}} < 0.5$ then it is "under-served". It is assumed that the initial value of the filling level is 0.5.

Flexibility value curves specify flexibility values as functions of flexibility status. This is akin to storage values versus filling level for storage generators. An example of such a curve is shown in Figure 5. If the associated nodal price is lower than the flexibility value, then the load is served. This in turn increases the flexibility status.

2.4 Grid power flow

The power balance at each node requires that all generation minus all consumption equals the net power flow *out* of the node. Additionally, AC power flow in the grid obeys power flow equations, and is moreover constrained by capacity limits. These physical relationships and constraints can be formulated by a set of equations and inequalities. Together with a cost function which describes the total cost of generation, the problem of minimising the cost becomes an optimal power flow (OPF) problem. In our case we apply approximate this problem using only linear relationships, which gives a linear programming (LP) problem formulation. This must be solved time step by time step, where time steps are coupled due to the presence of storage.

DC branches (i.e. HVDC connections) are considered fully controllable such that powr flow is a free variable only constrained by the branch capacity.

The linearised power flow equations are obtained as follows.

Assume there are N buses, enumerated from 1 to N. Denote branches by double indices referring to endpoint buses. The impedance on the branch between node i and j is thus written $z_{ij} = r_{ij} + jx_{ij}$, where $i, j \in \{1, 2, \ldots, N\}$. From these impedances , the bus admittance matrix Y may be constructed:

$$Y_{ij} = G_{ij} + jB_{ij} = \begin{cases} y_{ii} + \sum_{n \neq i} y_{ni} & i = j, \\ -y_{ij} & i \neq j, \end{cases} \tag{10}$$

where $y_{ij} = g_{ij} + jb_{ij} = \frac{1}{z_{ij}}$ is the admittance of the branch between i and j.

The AC power flow equations are

$$\begin{aligned} P_k &= \sum_{J=1}^{N} |v_k||v_j|\Big(G_{kj}\cos(\theta_k-\theta_j) + B_{kj}\sin(\theta_k-\theta_j)\Big), \\ Q_k &= \sum_{J=1}^{N} |v_k||v_j|\Big(G_{kj}\sin(\theta_k-\theta_j) - B_{kj}\cos(\theta_k-\theta_j)\Big). \end{aligned} \tag{11}$$

Here, P_k and Q_k are the net active and reactive power flow *into* node i. To arrive at the linearised equations, the following approximations are made:

- phase angle differences are small, so $\sin(\theta_i - \theta_j) \approx \theta_i - \theta_j$, and $\cos(\theta_i - \theta_j) \approx 1$
- voltage deviations are small, so $|v| \approx 1$ in per units system
- branch resistance is small compared to reactance, so $z_{ij} \approx jx_{ij}$
- shunt reactances are small, so ignoring self-admittance $y_{ii} \approx 0$

The last two approximations imply that $Y_{ij} \approx jB_{ij}$. The reactive power flow equation becomes trivial, and the power flow equations reduce to

$$P_k = \sum_{j=1}^{N} B_{kj}(\theta_k - \theta_j). \tag{12}$$

This is the linearised power flow equation. On matrix form it can be written

$$\Delta P = B'\Delta\theta, \tag{13}$$

where ΔP is the net power into a node (i.e. production minus consumption), and $\Delta\theta$ is the angle relative to the angle of a chosen reference node. The B' matrix is related to the B matrix.

The net power flow into a node $\Delta P = [P_k]$ is the sum of generation, demand, load shedding and hvdc power inflow:

$$P_k = \sum_{j=1}^{N_{\text{gen}}} P_j^{\text{gen}} - \sum_{j=1}^{N_{\text{pump}}} P_j^{\text{pump}} + P^{\text{shed}} + \sum_{j=1}^{N_{\text{dc}}} P_j^{\text{dc}} - \sum_{j=1}^{N_{\text{cons}}} P_j^{\text{cons}}, \tag{14}$$

where P_j^{gen} is generator output, N_{gen} is number of generators at the given node, P_j^{pump} is pump demand, N_{pump} is number of pumps at the given node, P^{shed} is amount of load shedding (unfulfilled demand), P_j^{dc} is inflow on DC branches (positive or negative), N_{dc} is number of DC branches connected to the given node, P_j^{cons} is consumer demand, and N_{cons} is number of consumers at the given node.

Power flow P_B on each branch is computed by the matrix equation

$$P_B = (DA)\Delta\theta, \tag{15}$$

where D is a diagonal matrix with elements given by the branch susceptance $D_{mm} = -b_m$, and A is the node–arc incidence matrix.

3 Optimisation problem

A linear objective function is used in order to ensure fast optimisation that converges, with the practical benefit that it also requires fewer input parameters.

The set of variables to be determined by the optimisation are

$$X = \{P_g^{\text{gen}}, P_p^{\text{pump}}, P_f^{\text{flex}}, P_n^{\text{shed}}, \theta_n, P_j\}, \tag{16}$$

where $g \in \mathcal{G}$, the set of generators; $p \in \mathcal{P}$, the set of pumps; $f \in \mathcal{F}$, the set of flexible loads; $n \in \mathcal{N}$, the set of nodes. $j \in \mathcal{B}$, the set of AC and DC branches.

The objective of the optimisation is expressed in terms of an objective function, which in our case is

$$\begin{aligned} F = &\sum_{g\in\mathcal{G}} c_g^{\text{gen}} P_g^{\text{gen}} - \sum_{p\in\mathcal{P}} c_p^{\text{pump}} P_p^{\text{pump}} \\ &- \sum_{f\in\mathcal{F}} c_f^{\text{flex}} P_c^{\text{flex}} + \sum_{n\in\mathcal{N}} c^{\text{shed}} P_n^{\text{shed}}, \end{aligned} \tag{17}$$

where c_g is the cost of generator g, c_p^{pump} is the cost of pump p, c_f^{flex} is the cost of flexible load p, and c^{shed} is the fixed cost of load shedding. As discussed in Section **??**, these cost parameters are determined by the fuel price for generators without storage, and by storage values in the other cases. The negative sign in front of pumping and flexible load means that increasing their value reduces the objective function. However, the energy balance constraint (see below) implies that power for pumping or flexible load must be compensated by generation elsewhere. So whether it is beneficial therefore depends on the cost of that alternative generation.

The variables (16) are not free, but constrained through upper and lower bounds, and through equations expressing relationships between them. Referring to these constraints as C_m, the optimisation problem is formulated in the standard Linear Programming (LP) form

$$\min F = \min \sum c_i X_i \quad \text{such that} \quad \{C_1, \ldots C_6\}. \tag{18}$$

This must be solved time step by time step, where time steps are coupled due to the presence of storage. The various constraints are now described in more detail.

The *first* set of constraints state that power flow on branches is constrained by their capacity limits:

$$C_1: \quad -P_j^{\max} \le P_j \le P_j^{\max} \tag{19}$$

where j refers to AC and DC branches with limited capacity.

The *second* set of constraints state that the power generation at generators is limited by lower and upper bounds, most notably the generation capacity and available power as described in Section **??**:

$$C_2: \quad P_g^{\min} \le P_g^{\text{gen}} \le P_g^{\text{limit}}, \tag{20}$$

where g refers to all generators.

The *third* set of constraints state that the pumping is limited by the pump capacity

$$C_3: \quad 0 \le P_p^{\text{pump}} \le P_p^{\text{pump,max}}, \tag{21}$$

where p refers to all pumps.

The *fourth* set of constraints state that the flexible load is limited by the maximum demand

$$C_4: \quad 0 \le P_f^{\text{flex}} \le P_f^{\text{flex,max}}, \tag{22}$$

where f refers to all flexible loads.

The *fifth* set of constraints express the condition of power balance at each node, which requires that net power injection at a node equals the net AC power flow out of the node. Net power injection is given as generated power minus demand, pumping and load shed plus power inflow via DC connections, which are controllable and free variables in the optimisation. Flow in the AC grid, however is determined by grid impedances by means of non-linear power flow equations. In order to formulate the problem as a linear optimisation problem, an approximate version of these equations is applied. To arrive at the linearised equations, often referred to as the DC power flow equations, the following assumptions are made: 1) phase angle differences are small; 2) voltage deviations are small; 3) branch resistance is small compared to reactance; 4) shunt reactances are small, so self-admittances can be ignored. With these assumptions the AC power flow equations reduce to the linear equations

$$C_5: \quad \mathbf{P}^{\text{node}} = \mathbf{B}'\boldsymbol{\Theta}, \tag{23}$$

where $\boldsymbol{\Theta}$ is a vector of voltage angles, $\mathbf{B}'$ is the conductance matrix, and $\mathbf{P}^{\text{node}}$ is a vector of net power injections into all nodes. The conductance matrix $\mathbf{B}'$ is the imaginary part of the bus admittance matrix,

which are the same with the approximations given above. The $\mathbf{P}^{\text{node}}$ vector elements are given as

$$P_k^{\text{node}} = \sum_{j \in \mathcal{G}_k} P_j^{\text{gen}} - \sum_{j \in \mathcal{P}_k} P_j^{\text{pump}} - \sum_{j \in \mathcal{C}_k} P_j^{\text{cons}} + P_k^{\text{shed}} + \sum_{j \in \mathcal{D}_k} P_j^{\text{dc}}, \tag{24}$$

where P_j^{gen} is generator output, $\mathcal{G}_k$ is the set of generators at node k, P_j^{pump} is pump demand, $\mathcal{P}_k$ is the set of pumps at node k, P_k^{shed} is amount of load shedding at node k, P_j^{dc} is inflow on DC branches (positive or negative), $\mathcal{D}_k$ is the set of DC branches connected to node k, P_j^{cons} is consumer demand (fixed *and* flexible), and $\mathcal{C}_k$ is the set of loads at node k.

The *sixth* set of constraints express the relationship between power flow on branches and nodal voltage angle differences. In the linear approximation, power flow $\mathbf{P}^{\text{ac}}$ on AC branches is related to nodal voltage angles as expressed by the equation

$$C_6: \quad \mathbf{P}^{\text{ac}} = \mathbf{DA\Theta}, \tag{25}$$

where $\mathbf{D}$ is a diagonal matrix with elements given by the branch reactance $D_{mm} = -\frac{1}{x_m}$, and $\mathbf{A}$ is the node–branch incidence matrix describing the network topology.

The *seventh* constraint specifies the reference node and its voltage angle,

$$C_7: \quad \theta_0 = 0. \tag{26}$$

Since these are arbitrary and don't influence the results, the reference is chosen such that the zeroth node has zero voltage angle.

4 Scenario generation tool

In order to simplify the process of generating scenarios with differeng generation mix and demand, the package includes a scenario module that can be used to save the loaded grid model to a scenario file. This is a summary file that includes demand per area, generation capacities per type and area etc. Exporting the grid model to a scenario file can be very useful for checking that the dataset is consistent with the scenario being studied.

The key functions are :

- powergama.scenarios.saveScenario(...)
- powergama.scenarios.newScenario(...)

4.1 Saving grid model to scenario file

To load an existing grid model and export a scenario file ("scenario.csv"'), run code similar to the following:

```
>>>import powergama
>>>import powergama.scenarios

>>>gridmodel = powergama.GridData()
>>>gridmodel.readGridData(nodes="nodes.csv",
                          ac_branches="branches.csv",
                          dc_branches="hvdc.csv",
                          generators="generators.csv",
                          consumers="consumers.csv")
>>>gridmodel.readProfileData(filename="profiles.csv",
                storagevalue_filling="profiles_storval_filling.csv",
                storagevalue_time="profiles_storval_time.csv",
                timerange=range(0,8760),
                timedelta=1.0)
```

```
>>>powergama.scenarios.saveScenario(gridmodel,
                                    scenario_file= "scenario.csv")
```

4.2 Create modified dataset using scenario file

In order to create a scenario file, the simplest is to to save an existing grid model to scenario file as shown above. Then, it can be open in a spreadsheet editor and modified according to the specifications of the new scenario. Values that should not be modified should be left blank. Irrelevant rows can be removed. In general, the newScenario function only modifies data where information is provided in the scenario file.

Be careful with profile reference data, as the output created by saveScenario will join together all references present for each country, and cannot be used directly when creating new scenarios. The information may be useful in validating the dataset, but is not useful for creating new scenarios. If a single reference is used for the country (e.g. all wind generators in a country), then it is ok to include this (e.g. demand reference), but for generator inflow and storage value references, it may be necessary to modify these values directly in the data file (if modification is required). If no modifications are required, these rows should be removed. This concerns the following rows in the scenario file:

- demand_ref
- inflow_ref_¡gentype¿
- storval_time_ref _¡gentype¿
- storval_filling_ref_¡gentype¿

Once a modified scenario file has been created ("scenario_new.csv"), run code similar to the following in order to create new input data files:

```
>>>import powergama
>>>import powergama.scenarios

>>>gridmodel = powergama.GridData()
>>>gridmodel.readGridData(nodes="nodes.csv",
                          ac_branches="branches.csv",
                          dc_branches="hvdc.csv",
                          generators="generators.csv",
                          consumers="consumers.csv")
>>>gridmodel.readProfileData(filename="profiles.csv",
                storagevalue_filling="profiles_storval_filling.csv",
                storagevalue_time="profiles_storval_time.csv",
                timerange=range(0,8760),
                timedelta=1.0)

>>>powergama.scenarios.newScenario(gridmodel,
                                   scenario\_file="scenario\_new.csv",
                                   new_file_prefix="new_")
```

The new input files will have the same names as the original, but with the prefix "new_", e.g. "new_nodes..csv".

Now, you can run a new simulation with these new input files instead of the original ones.

5 Input data formats

Input files are comma separated text files (CSV), with a comma as delimiter and period as the decimal symbol. The first line in the files contains a header, with unique keys associated with each column. The ordering of columns is arbitrary. Keys are case sensitive and should be all lower case.

5.1 Grid data

There are 5 input files associated with nodes, AC branches, DC branches, consumers and generators. These files contains references to additional files which has information about normalised storage values, energy inflow profiles and power demand profiles. The reference identifier (integer number or string) in the generator and consumer files should match an identifier in the relevant storage value or profile files.

5.1.1 Nodes

Nodes need to have unique identifier string. Area information is used for scenario generation (preprosessing), and for plotting and presentation of results. Latitude and longitude information is only used for plotting the grid on a map.

column key	description	type	units
"id"	Unique string identifier	string	
"lat"	Latitude	float	degrees
"lon"	Longitude	float	degrees
"area"	Area/country code	string	

5.1.2 AC Branches

Branches have from and to references that must match a node identifier in the list of nodes. Impedance should be given as per unit system with the the base power being the global one (powergama.constants.baseS)

column key	description	type	units
"node_from"	Node identifier	string	
"node_to"	Node identifier	string	
"reactance"	Reactance	float	p.u
"resistance"	Resistance (OPT)	float	p.u.
"capacity"	Capacity	float	MW

5.1.3 DC Branches

DC branches have from and to references that must match a node identifier in the list of nodes.

column key	description	type	units
"node_from"	Node identifier	string	
"node_to"	Node identifier	string	
"capacity"	Capacity	float	MW

5.1.4 Consumers

Consumers are loads connected to nodes. There may be any number of consumers per node, although zero or one is typical.

`demand_avg` gives the average demand, which is easily computed from the annual demand if necessary. `demand_ref` gives the name of the demand profile which gives the variation over time. Demand profiles should be normalised and have an annual average of 1.

column key	description	type	units
"node"	Node identifier	string	
"demand_avg"	Average demand	float	MW
"demand_ref"	Profile reference	string	
"flex_fraction"	Fraction of demand which is flexible (OPT)	float	
"flex_on_off"	Flexibility on/off ratio (OPT)	float	
"flex_storage"	Maximum flexibility (OPT)	float	MWh
"flex_storval_filling"	Profile ref, storage value filling dependence (OPT)	string	
"flex_storval_time"	Profile ref, storage value time dependence (OPT)	string	
"flex_basevalue"	Base storage value (OPT)	string	

5.1.5 Generators

Generators are the most complex data structure and require the most input data. The three columns related to pumping only need to be filled out if the pumping capacity is non-zero.

column key	description	type	units
"node"	Node identifier	string	
"desc"	Description or name	string	
"type"	Generator type	string	
"pmax"	Maximum production	float	MW
"pmin"	Minimum production	float	MW
"fuelcost"	Cost of generation	float	€/MWh
"inflow_fac"	Inflow factor	float	
"inflow_ref"	Inflow profile reference	string	
"storage_cap"	Storage capacity	float	MWh
"storage_price"	Base for storage value	float	€/MWh
"storval_filling_ref"	Profile ref, storage value filling level dependence	string	
"storval_time_ref"	Profile ref, storage value time dependence	string	
"storage_ini"	Initial storage filling level	float	1
"pump_cap"	Pumping capacity (OPT)	float	MW
"pump_efficiency"	Pumping efficiency (OPT)	float	
"pump_deadband"	Pumping price dead-band (OPT)	float	€/MWh

`node` is the string identifier of the node where the generator is connected. There may be any number of generators per node. `pmax` is the maximum power production, i.e. the generator capacity `pmin` is the minimum power production. This is normally zero, but may be nonzero for certain generator types such as nuclear power generators. `fuelcost` is the cost of generation. For generators without storage, the marginal cost is set equal to this value value. `storage_price` is the the base value for storage generator's storage values. It sets the absolute scale in the storage value calculation. `storage_capacity` is the capacity of the storage system. This is usually relevant only for hydro power and solar CSP. `storagevalue_ref` is the string identifier of the associated storage value table to be used for this generator/storage system `storage_init` is the initial relative filling level of the storage. `inflow_fac` is the inflow factor. `inflow_ref` is the string identifier of the associated inflow profile.

Power inflow at a given timestep t is computed according to

$$P_{\text{inflow}}(t) = P_{\max} \times \text{inflow factor} \times \text{profile value}(t). \tag{27}$$

In case the annual inflow is known, the inflow factor can be expressed by integrating the above equation, giving

$$\text{inflow factor} = \frac{\text{annual inflow}}{8760\ \text{h} \times P_{\max} \times \text{avg(profile value)}}. \tag{28}$$

There are two typical ways to use inflow factor and inflow profile:

- Normalised inflow profile with *maximum* value = 1: profile gives power output per installed capacity, with average value equal to the capacity factor of the generator. In this case, `inflow_factor` should be approximately 1, larger for good sites and smaller for bad sites. If inflow factor is larger than one, then at times $P_{\text{inflow}} > P_{\max}$.
- Normalised inflow profile with *average* value = 1: `inflow_factor` is equal to capacity factor, i.e. average inflow divided by generator capacity. Typical capacity factors are 0.5 for a large hydro storage system, 0.25 for wind power, and 0.22 for solar PV.

It is important to keep in mind that if the generator capacity is upgraded without the energy inflow changing (which may be relevant if there is storage), the inflow factor must be reduced correspondingly.

If fine resolution is not needed, many generators may use the same profile, but with different inflow factors to get representative capacity factor.

5.2 Time dependence of power consumption, power inflow and storage values

The following quantities vary with time:

- Generator power inflow (wind, solar radiation, rain)
- Consumer load (power demand)
- Storage values

For these, there are two field in the input data, one parameter that gives the absolute scale, and a reference to a normalised profile which entails the time profile. Multiplied together these give the absolute variation over time, as expressed e.g. in Eq. (27) The reason for this splitting between absolute scale and normalised profile is to enable multiple references to the same profile (e.g. normalised profile for demand may be the same for all consumers within an area), and to simplify the task of creating scenarios by scaling up or down the absolute scale without the need to change the profile time series.

Power inflow is given by weather conditions. Hydro has mainly a seasonal profile, whereas wind and solar varies from hour to hour. Solar has a characteristic daily profile with no production in dark hours. As stated previously, there are two alternative ways to specify inflow profile and absolute scale (inflow factor) are used: 1) The profile is normalised to give power inflow per installed capacity (with average value representing the capacity factor), and absolute scale is nominally equal to one; 2) The profile is normalised to have an average value of unity, and absolute scale represents the capacity factor.

column key	description	type	units
identifier1	values type 1	float	MW
identifier2	values type 2	float	MW
...			

There is one row per time step.

5.3 Storage values

There are two dependencies:

- Filling level
- Time of year and time of day

All in all, the storage values are computed according to

$$\text{storage value}(f,t) = \text{base value} \times \text{filling level profile}(f) \times \text{time profile}(t), \tag{29}$$

where f is the relative filling level, and t is the timestep.

Time dependence of storage values reflect the time dependence of the associated inflow, and is therefore quite different for hydro (seasonal variation) and CSP (daily variation). This depenency is given in the same format as for inflow and consumption, see above.

Storage value dependence on filling level is specified as follows:

column key	description	type	units
identifier1	values type 1	float	€/MWh
identifier2	values type 2	float	€/MWh
...			

There is one row per percentile (filling level)

6 Output data

6.1 Optimal solution

The primary result are values for the cost function (total cost of generation) and values for all variables for each timestep in the simulation. The variables are

- power generation for each generator
- voltage angles at nodes
- power flow on AC branches (actually derived from voltage angles)
- power flow on DC branches
- load shedding at each node

Derived quantities include

- storage level and marginal price for generators with storage
- spilled power inflow (e.g. constrained/curtailed wind power)

6.2 Sensitivities

Sensitivities are computed for the following variables:

- AC branch capacity
- DC branch capacity
- Power demand at each node

These sensitivities say how much the total generation cost would increase if branch capacity or power demand at a given branch or node were to increase by one unit. This is useful for identifying grid bottlenecks and nodal power prices.

6.3 Further analysis of results

Examples of interesting analyses that can be addressed using PowerGAMA are

- Identification of grid bottlenecks. This is relevant for existing bottlenecks, but even more so with future scenarios with new generators installed, e.g. large amounts of renewable generation. Assessment of ot benefits by reinforcing certain connections, or adding more lines.
- Identification of the potential of the grid and power system to absorb large amounts of renewable generation. How much new capacity of wind and solar power can be introduced without problems
- Estimation of generation mix

It should be noted that PowerGAMA does not include any power market subtleties (such as start-up costs, forecast errors, unit commitments) and as such will tend to overestimate the ability to accomodate large amounts of variable renewable energy. Essentially it assumes a perfect marking based on nodal pricing without barriers between different countries. This is naturally a gross oversimplification of the real power system, but gives nontheless very useful information to guide the planning of grid developments and to assess broadly the impacts of new generation and new interconnections.

6.4 Included plots and other result analysis functions

See the online powergama source code documentation for a complete overview of plotting functions and other functions to retreive simulation results for further analysis.

7 Power Grid Investment Module (PowerGIM)

The PowerGAMA includes a grid investment module for determining socio-economically beneficial grid investments (connections and generators). This module is documented separately.

However, an overview of the input data format is provided in the following.

7.1 PowerGIM grid data input

PowerGIM input have the same format as PowerGAMA data (see above), but some fields (columns) differ.

7.1.1 Nodes

Nodes need to have unique identifier string. Area information is used for scenario generation (preprosessing), and for plotting and presentation of results. Latitude and longitude information is only used for plotting the grid on a map.

column key	description	type	units
"id"	Unique string identifier	string	
"lat"	Latitude	float	degrees
"lon"	Longitude	float	degrees
"area"	Area/country code	string	
"existing"	Whether node already exists	boolean (0/1)	
"cost_scaling"	Cost scaling factor	float	
"type"	Node (cost) type	string	

7.1.2 Branches

Branches have from and to references that must match a node identifier in the list of nodes.

`expand` is 0 if no expansion should be considered, 1 if expansion in stage 1 should be considered, and 2 if expansion in stage 2 should be considered. (If expansion in both stage 1 and 2 should be considered, add a new branch with 0 existing capacity). `distance` may be left blank. Then distance is computed as the shortest distance between the associated nodes (based on lat/lon coordinates) `capacity2` is already decided additional branch capacity that will be added at stage two (optional input). It may be relevant to include this as an uncertain parameter in a stochastic optimisation.

column key	description	type	units
"node_from"	Node identifier	string	
"node_to"	Node identifier	string	
"capacity"	Existing capacity	float	MW
"capacity2"	Capacity added stage 2 (OPT)	float	MW
"expand"	Consider expansion	int (0,1,2)	
"distance"	Branch length (OPT)	float	km
"max_newCap"	Max new capacity (OPT)	float	km
"cost_scaling"	Cost scaling factor	float	
"type"	Branch (cost) type	string	

7.1.3 Consumers

Consumers are loads connected to nodes. There may be any number of consumers per node, although zero or one is typical.

`demand_avg` gives the average demand, which is easily computed from the annual demand if necessary. `demand_ref` gives the name of the demand profile which gives the variation over time. Demand profiles should be normalised and have an annual average of 1.

column key	description	type	units
"node"	Node identifier	string	
"demand_avg"	Average demand	float	MW
"demand_ref"	Profile reference	string	
"emission_cap"	Maximum CO2 emission allowed (OPT)	float	kg

7.1.4 Generators

Generators are the most complex data structure and require the most input data.

An average power constraint (`pavg`) is used to represent generators with large storage. `pavg`=0 means no constraint on average output is used (no storage constraint). Already decided increase in generator capacity in stage 2 may be specified with `pmax2` parameter.

column key	description	type	units
"node"	Node identifier	string	
"desc"	Description or name (OPT)	string	
"type"	Generator type	string	
"pmax"	Generator capacity	float	MW
"pmax2"	Generator capacity stage 2 (OPT)	float	MW
"pmin"	Minimum production	float	MW
"expand"	Consider capacity expansion	int (0,1,2)	
"fuelcost"	Cost of generation	float	€/MWh
"fuelcost_ref"	Cost profile	string	
"inflow_fac"	Inflow factor	float	
"inflow_ref"	Inflow profile reference	string	
"pavg"	Maximum average power (OPT)	float	MW
"p_maxNew"	Maximum new capacity (OPT)	float	MW
"cost_scaling"	Cost scaling factor (OPT)	float	

7.2 PowerGIM parameter data input

In addition to the network data, PowerGIM requires additional parameters (mostly cost parameters) that are specified in a separate XML file.

The format of this file may be seen from the examples.

8 Examples

Examples are provided on the web site [5]:
`https://bitbucket.org/harald_g_svendsen/powergama/wiki/Home`

References

[1] M. Korpås, L. Warland, J. O. G. Tande, K. Uhlen, K. Purchala, S. Wagemans, TradeWind D3.2: Grid modelling and power system data, Tech. Rep. TR F6604, SINTEF Energy Research (2007).

[2] D. Huertas-Hernando, H. G. Svendsen, L. Warland, T. Trötscher, M. Korpås, Analysis of offshore grid alternatives for north sea offshore wind farms using a fow-based market model, in: Proceedings of the European Energy Markets Conference (EEM), Madrid, SPAIN, 2010, doi:10.1109/EEM.2010.5558725.

[3] O. B. Fosso, A. Gjelsvik, A. Haugstad, B. Mo, I. Wangensteen, Generation scheduling in a deregulated system: The norwegian case, IEEE Transaction on Power Systems 14 (1) (1999) 75–81. `doi:10.1109/59.744487`.

[4] O. Wolfgang, A. Haugstad, B. Mo, A. Gjelsvik, I. Wangensteen, G. Doorman, Hydro reservoir handling in norway before and after deregulation, Energy 34 (10) (2009) 1642–1651, doi:10.1016/j.energy.2009.07.025.

[5] H. G. Svendsen, Powergama user guide, `https://bitbucket.org/harald_g_svendsen/powergama/wiki` (accessed 2017-01-03) (2014).