RES-BESS size and dispatch optimisation engine - version0.0.1

Luka Smajila

Last Update: January 29, 2025

1 Introduction

The aim of this document is to provide an overview of the version1 of the LCA-SESS tool, which is developed to optimise the dispatch of a PV-BESS system in the context of Decision Support via Techno-Economic and Techno-Environmental analysis and optimisation. Figure 1 shows the overall methodology and serves to characterise the behavior of the model. This is further described in subsequent sections.

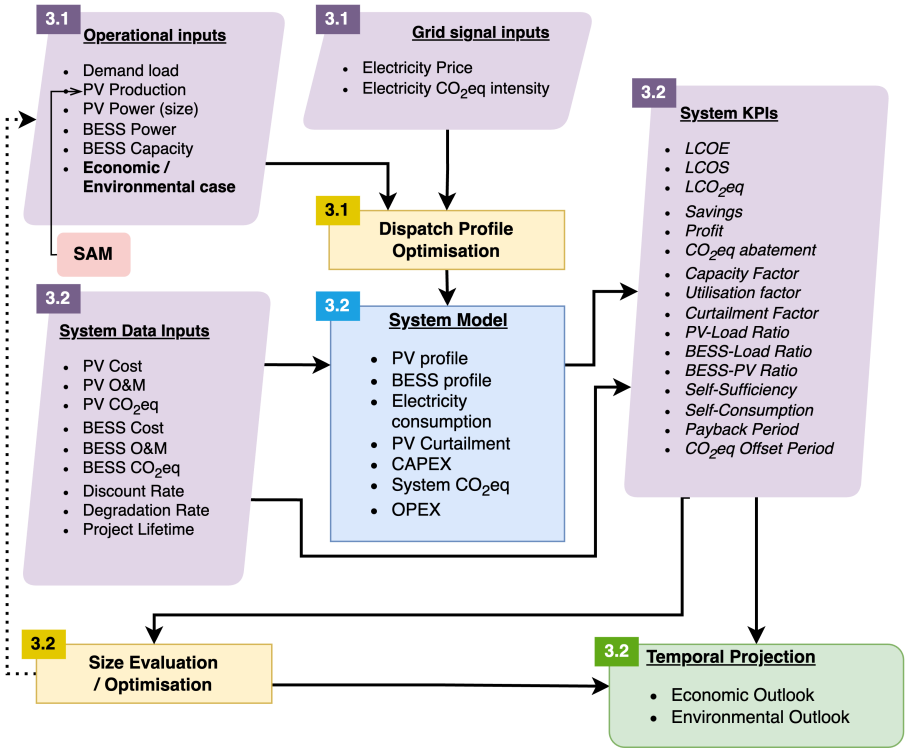

Figure 1: Model *version1* - Methodological representation of the model for a PV-BESS hybrid system

The next part of the document describes how different parts of the model perform and what they can be used for (mainly components 3.1 - Dispatch Optimisation Algorithm and 3.2 - Sizing Optimisation via Parametric Exhaustive-Search algorithm.

Note to self:
Next, describe the requirements for installation of the model (which python and optimisation engine to install [GLPK]), give an overview of the dispatch optimisation and the subsequent size variation, and (later) size optimiser that takes into consideration the KPIs that we will specify. For Reporting purposes, this section can be omitted

2 Installation, Tools & Environments

2.1 Installation

The RES-BESS model is built in Python.

1. Install Python 3.XX from the python.org (what is important is to have Python version 3)
2. Install dependencies (Python packages)

```
pip install -r requirements.txt
```

3. install the GLPK solver
4. Run code!

2.2 Recommended tools

I recommend using VisulStudio Code (link), primarily as it allows for editing multiple script files simultaneously and comes with an integrated terminal. Of course, other tools can be used at the user's discretion.

2.3 Making virtual environments in Python (Terminal)

In order to minimise so-called "project fatigue" and interdependency problems, I recommend creating a virtual environment for the model (not included in the online repository) which allows us to install and activate the relevant Python packages without affecting the "clean" version of the system python installation.
Below is a simple walkthrough of the installation and setting up of the first virtual environment and creating a Jupyter kernel, so the virtual environment can be used with Jupyter notebooks.

2.3.1 Virtualenv installation

- Option 1: using virtualenv
  1. pip install virtualenv in terminal (link to documentation)

```
python -m pip install virtualenv
```

  2. In terminal run:

```
python -m virtualenv <my_env_name>
```

  3. Activate the environment (in terminal)

```
source <my_env_name>\Scripts\activate
```

     if MacOS/Linux:

```
source <my_env_name>/bin/activate
```

  4. Install packages, run code in terminal, etc.
  5. Deactivate the environment when done with work (in terminal)

```
deactivate
```

- Option 2: using venv (comes included in Python)
  1. In Terminal:

```
python -m venv <my_env_name>
```

2. Activate (in terminal)

```
source <my_env_name>\Scripts\activate
```

   if MacOS/Linux:

```
source <my_env_name>/bin/activate
```

3. Install packages, run code in terminal, etc.
4. Deactivate the environment when done with work (in terminal)

```
deactivate
```

Some online guides to help: RealPython, Python-venv, Python-use of pip and venv, Medium.com - virtual environments

2.3.2 Creating a Jupyter Kernel

Next, we can create a Jupyter Kernel so that we will easily use the virtual environment inside a jupyter notebook and run our code (good for sequential code testing, analysis and having one file to run the rest of the model)

1. Activate the virtual environment (in terminal)

```
source <my_env_name>\Scripts\activate       --Windows
//
source <my_env_name>/bin/activate           --MacOS/Linux
```

2. install Jupyter, Notebook, iPython, and kernel (in terminal)

```
python -m pip install jupyter notebook ipython kernel --upgrade
```

3. install kernel (creates a kernel from our environment)

```
ipython kernel install --user --name=<my_env_name>
```

4. End, now our virtual environment should be visible under kernels in the jupyter notebook interface

   Online guides: Python - GeeksForGeeks, IPython

3 RES-BESS Model

While the first iteration of the model explores a PV-BESS system (as described in PaperA), the model can represent any kind of behind-the-meter power production, most commonly from renewable energy sources (Figure 2).

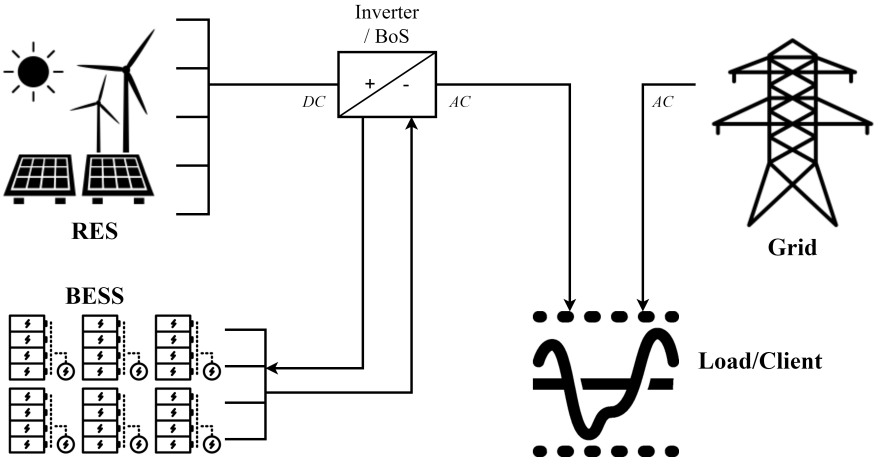

Figure 2: Representation of a hybrid RES-BESS system

This hybrid system is connected to the Load Demand and to the (local) energy grid. This is to ensure the energy demand of the load in the system is always fulfilled, either by utilising the RES-BESS system or by sourcing electricity from the grid. Additionally, the connection of the system to the grid allows for several business use-cases, such as renewables firming, increasing the BESS utilisation factors, and energy arbitrage with the electricity grid. While this can be sufficiently representative of a behind-the-meter system, future development of the model will focus on extending the functionality of operational model and business cases for Energy Storage that the model can characterise.

3.1 Dispatcher

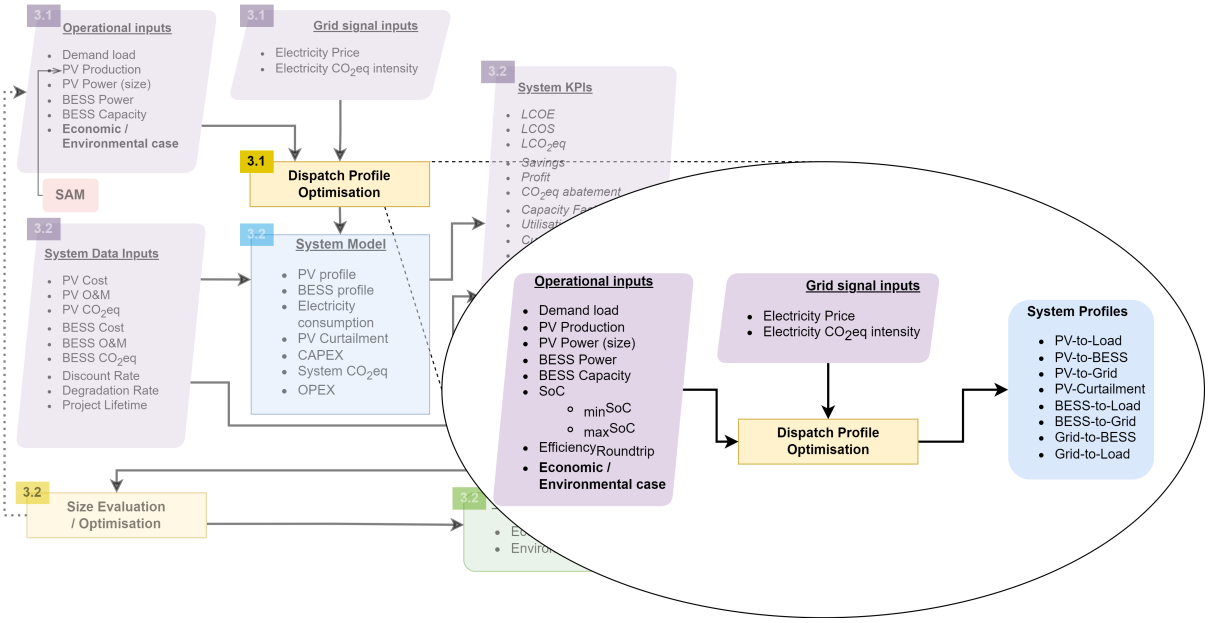

Figure 3: Model flow of the Dispatch Optimisation portion of the model

The dispatch optimization employs a Mixed Integer Linear Programming (MILP) algorithm, implemented with Pyomo in Python and solved using the GLPK solver. This hourly-based algorithm operates within a rolling 48-hour window to address yearly operational demand. The key operational inputs are the load demand, solar operational profile, and grid signal time-series in a 1-hour time-step. The grid signals represent the curve that the algorithm optimises for. In this framework, the grid signals can be the price of electricity (given in EUR/MWh) or the intensity of electricity production (given in CO_2eq/kWh) for each time step. BESS-specific parameters are used, such as the charging and discharging efficiency (Eff_{charge} and $Eff_{discharge}$, respectively), and the allowed minimum and maximum State-of-Charge levels (SoC^min and SoC^{max}, respectively). These parameters are further part of the operational constraints. In addition to the key operational inputs such as the Load Demand profile, RES production profile (in this case PV) and the selected grid signal that the dispatcher should optimise for (all visualised in Figure 2, the specified BESS power rating and energy capacity (MW and MWh, respectively) represent the available storage for energy in the dispatch model. This signal can be the electricity price, or the intensity of electricity production, depending on whether the dispatch is optimised for minimising economic cost, or minimising the attributed environmental footprint of our system/user demand. The dispatch optimization minimises the cost function, specified in Equation 1, where the first element, based on the grid signal, incorporates power flow and signal value, with units dependent on the optimisation goal. The first element in this equation represents the grid signal and includes the power flow component ($P_{Grid-to-Load}$) and the value of the grid signal at that iteration in the time series t (C_{grid}) which can store the value of the electricity price or the electricity intensity, depending on the optimisation approach and/or goal. This is represented by index i, which represents the optimisation approach. If the dispatch optimisation approach/goal is economic, factor C^i_{Grid} represents the price at t and is expressed in EUR/MWh. If the approach/goal is environmental, it represents the intensity of electricity in the grid at t and is expressed in gCO_2eq/kWh. Similarly, the factor $C_{PV+BESS}$ represents the price, or CO_2eq footprint of our system, again depending on the objective of the optimisation.

Factors $P_{Grid-to-Load}$, $P_{PV-to-Load}$, and $P_{BESS-to-Load}$ represent the power flows from the grid, PV, or BESS to the Load Demand at that iteration of t, respectively. $C^{Curtailment}_{penalty}$ is assigned to the power flow that is "waste" by curtailing excess PV production. This is to ensure the dispatch algorithm aims to minimise the excess amount of PV production. This value should be high enough to not affect the performance of the dispatch algorithm for all scenarios. Finally, the last element,

the same grid factor C_{grid} and the power flows between the BESS and the grid ($P_{BESS-to-Grid}$ and $P_{Grid-to-BESS}$), represents the profit that can be incurred from energy arbitrage (when it occurs), and has a negative sign since the objective of the dispatch algorithm is to minimise the value of the function.

$$F_i = min \sum_{t=0}^{48} + \begin{cases} C^i_{grid}[t] * P_{Grid-to-Load}[t] \\ + C^i_{PV+BESS}[t] * \sum (P_{PV-to-Load}[t] + P_{BESS-to-Load}[t]) \\ + C^{Curtailment}_{penalty} * P_{Curtailment}[t] \\ - C^i_{grid}[t] * ((P_{BESS-to-Grid}[t] + P_{PV-to-Grid}[t]) - P_{Grid-to-BESS}[t]) \end{cases} \tag{1}$$

The algorithm is constrained by equations that represent the operational properties of the different sub-systems. The primary constraint of satisfying the Load Demand. This is ensured by equation 2, where P_{Load} represents the total Load Demand at a point in time t, which has to be satisfied by the different available power flows to the load from the sub-systems. $P_{PV-to-Load}$ is the power flow from the connected solar production, $P_{BESS-to-Load}$ is the flow of energy stored in the BESS, and $P_{Grid-to-Load}$ is the power flow of electricity purchased from the grid.

$$P_{Load}[t] = P_{PV-to-Load}[t] + P_{BESS-to-Load}[t] + P_{Grid}[t] \tag{2}$$

Further constraints are specified for the power coming from the PV production (equation 3, to ensure the algorithm accounts for all the power production and its correct use. $P_{PVproduction}$ is the total produced power at a point in time t, with $P_{PV-to-Load}$, $P_{PV-to-BESS}$ and $P_{PV-to-Grid}$ represent power flows to the different parts of the system. $P_{Curtailment}$ is the excess power that is curtailed.

$$P_{PVproduction}[t] = \frac{P_{PV-to-Load}[t] + P_{PV-to-BESS}[t] + P_{PV-to-Grid}[t]}{\xi_{inverter}} + P_{Curtailment}[t] \tag{3}$$

The BESS can be charged either from the PV production ($P_{PV-to-BESS}$) or the grid ($P_{Grid-to-BESS}$, depending on the business case and the state of switches (described in section **??**). This is reflected in equation 4. The losses in energy that occur in the charging and discharging process are included by the multiplication with the charging/discharging efficiencies (Eff_{charge} and $Eff_{discharge}$, respectively). Furthermore, self-discharge losses are neglected as the cycling of the system is assumed to be frequent enough for those losses to not have a measurable impact on the actual energy stored.

$$\begin{aligned} SoC[t] = & \; SoC[t-1] \\ & + (P_{PV-to-BESS}[t] + P_{Grid-to-BESS}[t] * \xi_{charge}) * \Delta t \\ & - \frac{P_{BESS-to-Load}[t] + P_{BESS-to-Grid}[t]}{\xi_{discharge}} * \Delta t \end{aligned} \tag{4}$$

Equation 5 constrains the amount of electricity that can be stored in BESS at any given point in time, by limiting the potential SoC level at a point in time t between SoC^{min} and SoC^{max}, which represent the minimum and maximum amount of energy capacity to be stored. These values are calculated as a percentage of the total energy capacity input of the BESS system, which is set at 10% for the minimum, and 90% for the maximum value. This is done so that a linear degradation rate of the

system can be assumed in the model, while it has been shown that the degradation rate of battery cells becomes non-linear at tail-ends of the SoC range.

$$SoC^{min} < SoC[t] < SoC^{max}; \qquad for\ any\ t \tag{5}$$

Lastly, to ensure the model represents a realistic BESS operation, two constraints are specified to ensure the BESS either charges or discharges at any given hour increment in the operation profile. This is described by equations 6, 7, and 8

$$P_{PV-to-BESS}[t] + P_{Grid-to-BESS}[t] <= Switch_{Charge} * P_{MaxCharge/DischargePower}; \qquad for\ any\ t \tag{6}$$

$$P_{BESS-to-Load}[t] + P_{BESS-to-Grid}[t] <= Switch_{Discharge} * P_{MaxCharge/DischargePower}; \qquad for\ any\ t \tag{7}$$

$$Switch_{Charge} + Switch_{Discharge} <= 1; \qquad for\ any\ t \tag{8}$$

3.1.1 Dispatch Optimisation Algorithm Output

Once the Dispatch optimisation is run, the output profiles can be analysed to represent the operation of the system. This is visualised in Figures 4, 5, and 6.

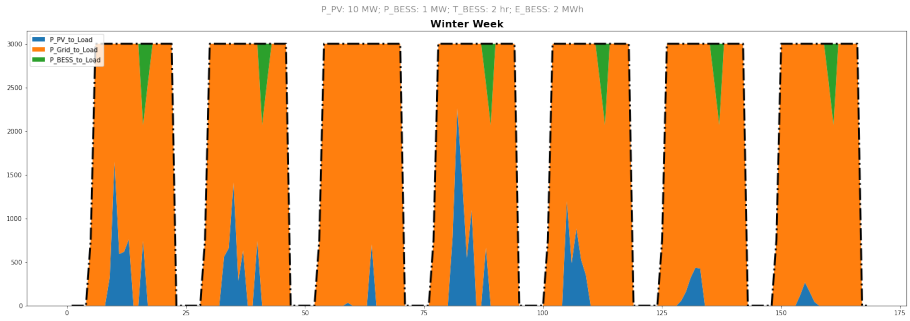

Figure 4: Load Fulfillment in a representative Winter Week

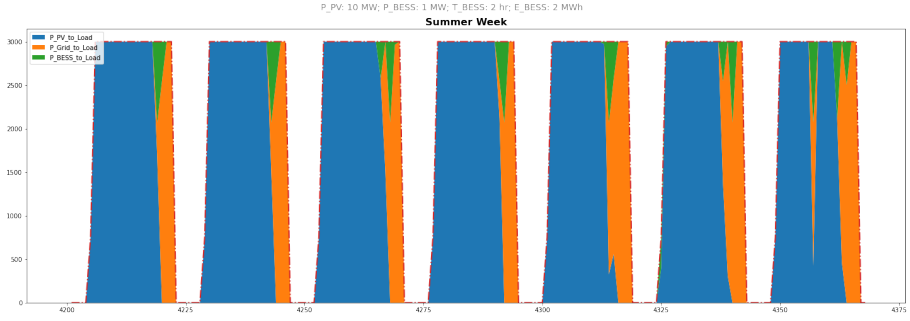

Figure 5: Load Fulfillment in a representative Summer Week

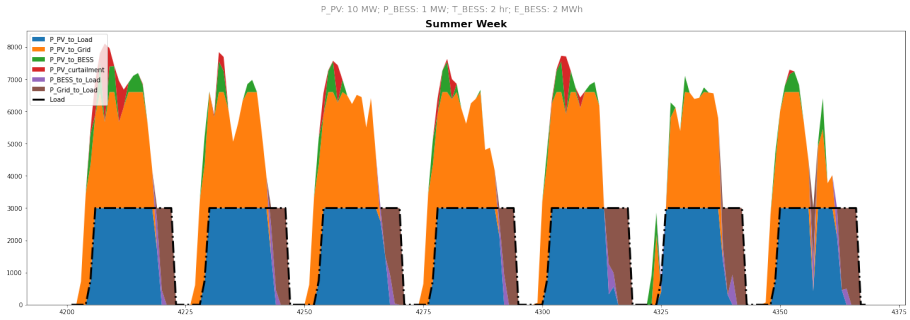

Figure 6: PV production and allocation in a representative Summer Week (In a winter week, PV production is less than load, so that figure would be equivalent to Figure 4

Similarly, the operation of BESS and its cycling can be analysed (Figures 7 and 8).

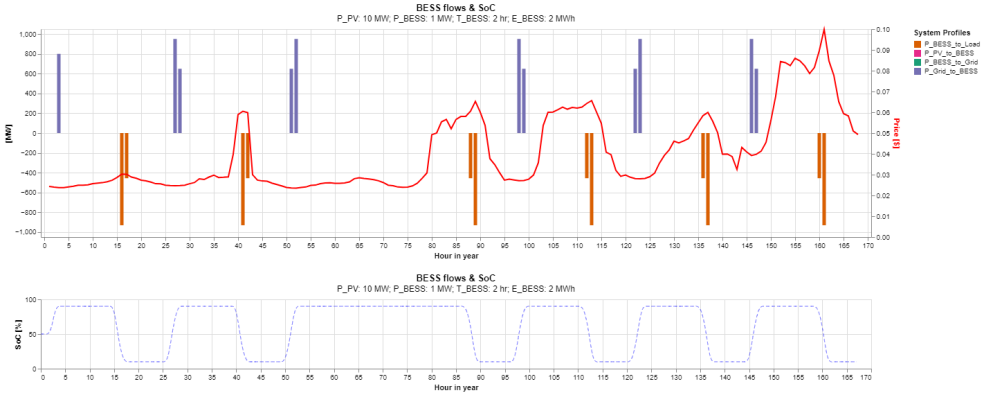

Figure 7: BESS operation in a representative Winter Week

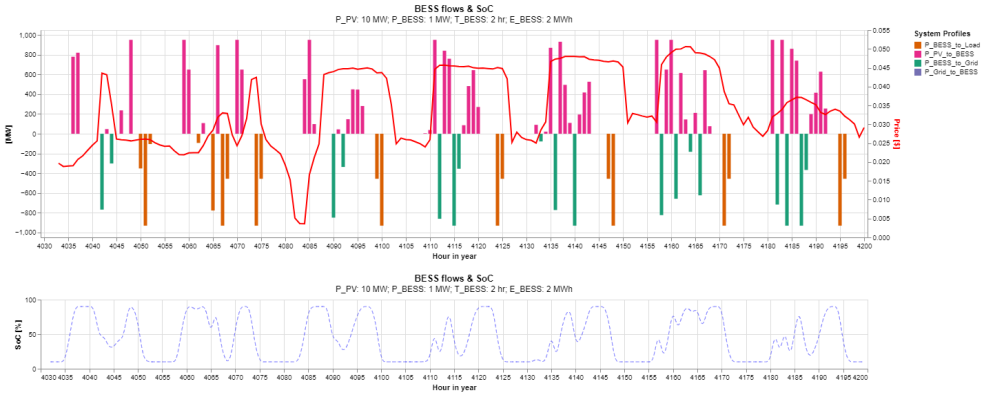

Figure 8: BESS operation in a representative Summer Week

3.2 Size optimiser

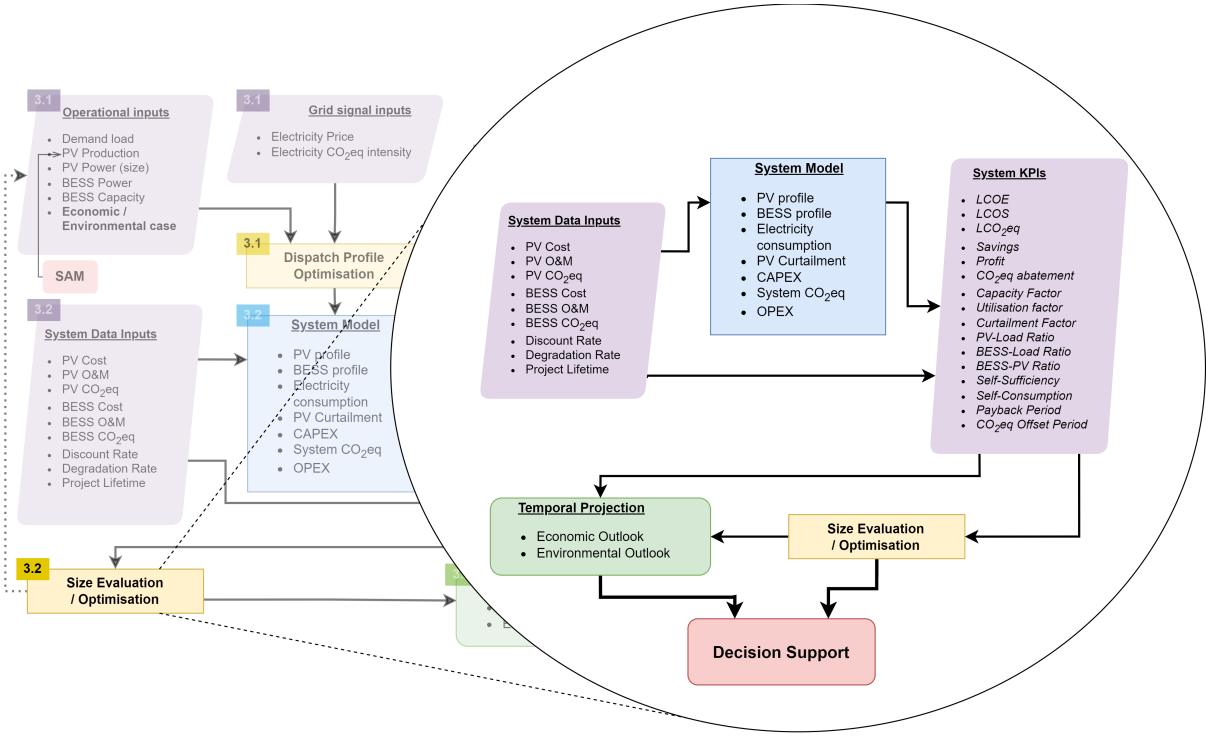

Figure 9: Model flow of the Sizing portion of the model

The system model uses the output hourly profile from the dispatch optimisation model and the system-related inputs (System Data input in Figure 9) to produce system-specific key performance indicators (KPIs). The main purpose of this model is to characterise the levelised cost and levelised direct CO_2eq impact of the system, and determine the associated business-case related savings/profit, potential CO_2eq abatement, and performance factors.
The CAPEX is calculated using equation 9, and includes the size and price of the PV system, the capacity and price of BESS (price of cells, nominally EUR/kWh), and the BOS components of the PV-BESS (including inverter), that are used to ensure the power delivery of the system as per the rated power. A similar approach is used for calculating the system-related OPEX (Equation 10).

$$\begin{aligned} CAPEX = & P_{PV} * Cost^{PV}_{Module} \\ & + P_{PV} * Cost^{PV-system}_{BoS} \\ & + E_{BESS} * Cost^{BESSCapacity}_{BESSCells} \\ & + P_{BESS} * Cost^{BESSPower}_{BoS} \end{aligned} \tag{9}$$

$$\begin{aligned} OPEX = & P_{PV} * Cost^{PV}_{O\&M\ per\ module} \\ & + P_{PV} * Cost^{PV-system}_{O\&M\ of\ BoS} \\ & + E_{BESS} * Cost^{BESSCapacity}_{O\&M\ per\ BESS\ Cell} \\ & + P_{BESS} * Cost^{BESSPower}_{O\&M\ of\ BoS} \end{aligned} \tag{10}$$

The CO_2eq production phase footprint is collected using the published LCA studies for a representative PV module and representative BESS. The CO_2eq equivalent of OPEX are sourced from published studies (**reference)** and the industry-reported assumptions (**NREL reference**). Similarly, associated End-of-Life costs and CO_2eq impacts, EoL_{Costs} and $EoL_{CO_2eq\ impacts}$, respectively, are sourced from

published studies and industry indications for value destruction & retention in PV modules and in BESS recycling processes.

These values are annualised (in the case of recurring costs such as O&M) and used to produce the levelised cost of energy (LCOE), with the annualised energy production/delivery profile provided by the dispatch optimisation algorithm. This is described in Equation **??**. For this analysis, two approaches were used to calculate the LCOE indicator (and the LCO2eq indicator). Firstly, equation 11 describes the levelised cost of the system as an investment case (investor perspective). Secondly, equation 12 describes the LCOE as the final cost to the behind-the-meter user/client.

$$LCOE_{Investor} = \frac{CAPEX + (OPEX_{O\&M} + OPEX^{BESS}_{EnergyProcurement}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}) + EoL_{Costs}}{\sum((E^{PV-BESS-to-Load}_{annual} + E^{PV-BESS-to-Grid}_{annual}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}))} \tag{11}$$

$$LCOE_{EndUser} = \frac{CAPEX + OPEX_{EndUser} * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}) + EoL_{Costs}}{\sum((E^{PV-BESS-to-Load}_{annual} + E^{Grid-to-Load}_{annual}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}))} \tag{12}$$

where $OPEX_{EndUser}$ is constructed with equation 13:

$$OPEX_{EndUser} = OPEX^{O\&M} + OPEX^{Load}_{EnergyProcurement} + OPEX^{BESS}_{EnergyProcurement} * \alpha_{BESS->Load} \tag{13}$$

Similarly to LCOE, the Lelised CO_2eq indicator is constructed to represent the levelised annualised footprint of the system. This is described with equations 14 and 15. *The associated CO_2eq contributions from O%M are neglected and assumed to equate to 0.

$$LCO_2eq_{Investor} = \frac{CO_2eq^{Production} + (OPEX^{CO_2eq}_{O\&M} + OPEX^{BESS}_{CO_2eqBurden}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}) + EoL_{Costs}}{\sum((E^{PV-BESS-to-Load}_{annual} + E^{PV-BESS-to-Grid}_{annual}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}))} \tag{14}$$

$$LCO_2eq_{EndUser} = \frac{CO_2eq^{Production} + OPEX^{CO_2eq}_{EndUser} * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}) + EoL_{Costs}}{\sum((E^{PV-BESS-to-Load}_{annual} + E^{Grid-to-Load}_{annual}) * \sum_{j=0}^{N}(\frac{1}{(1+i)^N}))} \tag{15}$$

where $OPEX^{CO_2eq}_{EndUser}$ is constructed with equation 16:

$$OPEX^{CO_2eq}_{EndUser} = OPEX^{CO_2eq}_{O\&M} + OPEX^{Load}_{CO_2eqBurden} + OPEX^{BESS}_{CO_2eqBurden} * \alpha_{BESS->Load} \tag{16}$$

In addition to the levelised economic and environmental indicators, several other performance factors are calculated by the system model. By annualising the energy flows, provided by the dispatch optimisation algorithm (described in section **??**) and the system demand that the algorithm sought to satisfy, factors such as self-sufficiency (Equation 17), self-consumption (Equation 18), and curtailment factor (Equation 19).

$$RenewableEnergySelf-Sufficiency = \frac{\sum_{j=0}^{N}(E_{PV-to-Load} + E_{BESS-to-Load} * \beta_{PV->BESS})}{\sum_{j=0}^{N} E_{LoadDemand}} * 100\% \tag{17}$$

$$Self-Consumption = \frac{\sum_{j=0}^{N}(E_{PV-to-Load} + E_{BESS-to-Load})}{\sum_{j=0}^{N} E_{SolarProduction}} * 100\% \tag{18}$$

$$CurtailmentFactor = \frac{\sum_{j=0}^{N} E_{PV-Curtailment}}{\sum_{j=0}^{N} E_{SolarProduction}} * 100\% \tag{19}$$

In addition, the system savings and the CO_2eq abatement are calculated via an analysis of the system profiles, output by the dispatch algorithm, shown in Figure **??**. By comparing the relative values of the $C^i_{Grid}[t]$ at each hourly time-step t and calculating the net energy flows from the PV-BESS system to the grid, representative cash and CO_2eq flows are determined at each time step and added for a

yearly total. This is described in Equation 20.

$$Savings/Abatement = \sum_{j=0}^{8760} \left(C^{i}_{Grid}[t] * \Delta P_{PV-BESS}[t] \right) \tag{20}$$
$$where \quad \Delta P_{PV-BESS}[t] = ((P_{PV-to-Load}[t] + P_{BESS-to-Load}[t]) - (P_{Grid-to-Load}[t]))$$

Continuing, the system payback period and CO_2eq offset periods are defined based on the annualised cash flows and CO_2eq displacement (avoided CO_2eq, respectively (Equations 21 & 22).

$$PaybackPeriod = \frac{CAPEX_{PV-BESS}}{Savings - Opex_{PV+BESS}} \tag{21}$$

$$OffsetPeriod = \frac{CO_2eq^{Production}_{BESS} + CO_2eq^{Production}_{PV}}{CO_2eq^{Avoided}_{scenario} - CO_2eq^{O\%M}_{PV+BESS}} \tag{22}$$

Optimal system size is determined for the three decision variables in the model - the size of the PV installation by the specified power rating P_{PV}, the capacity size of the BESS E_{BESS} and the power rating of the BESS P_{BESS}. A parametric analysis is performed based on the selected objectives, which can be chosen among the outputs of the system model (system KPIs in Figure **??**) and described in equations #**??** through #19.
The analysis aims to determine the trade-offs and optimal proposed solution on basis of selecting the system sizes that have the desired values of the value function:

$$\overline{F}(P_{PV}, P_{BESS}, E_{BESS}) = min(LCOE) \tag{23}$$

$$\overline{F}(P_{PV}, P_{BESS}, E_{BESS}) = max(SS) \tag{24}$$

Finally, once a proposed solution has been identified, temporal forecast for the lifetime of the system project and takes into account the related costs and added associated emissions of any sub-system replacement (both specified as System Data Inputs, described in Figure 1). This projection serves to evaluate the performance of multiple proposed solutions to the statu-quo (BAU scenario) and any location-specific economical and/or environmental/sustainability targets.

3.2.1 Sizing Model Output

The System and Sizing model output the values of KPIs for different system sizes (looped with the dispatch optimisation algorithm as described in Figure 1). Plotted values can be used to determine the trade-offs between objective KPIs (main goals of the decision-maker) and determine optimal size ranges, or even singular size, to best satisfy the objective.

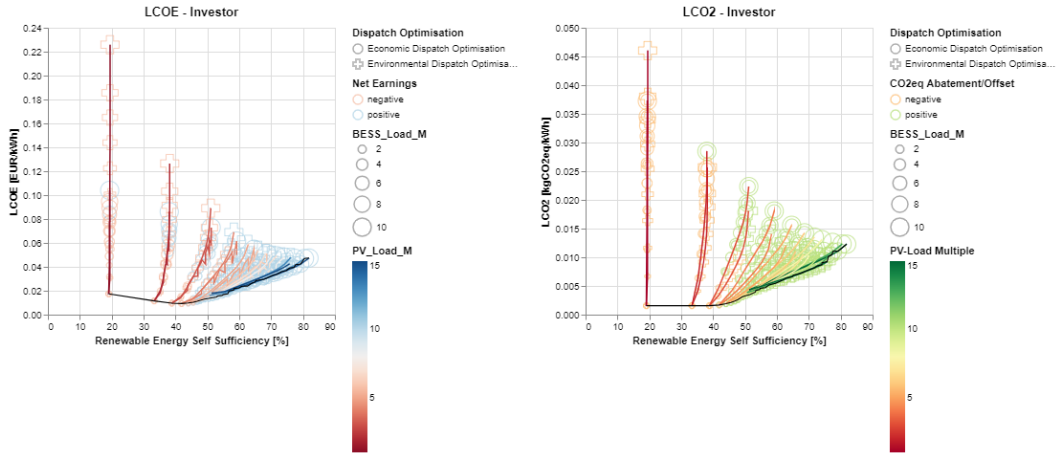

Figure 10: Optimality front (Bold Black Line) when optimising for LCOE and RESS for the investment case

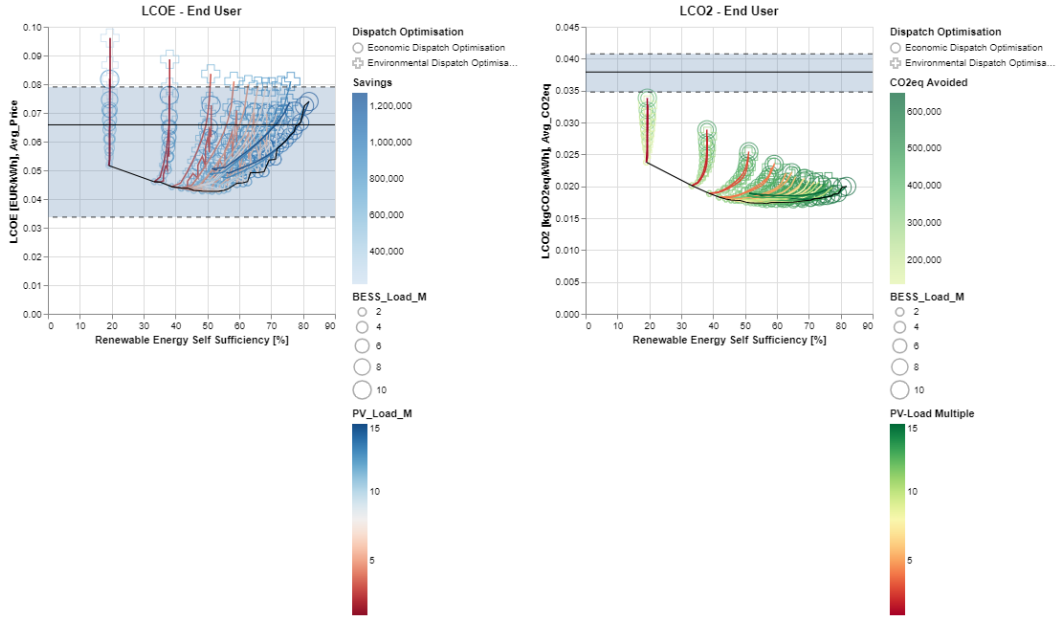

Figure 11: Optimality front (Bold Black Line) when optimising for LCOE and RESS for the user/client case

Furthermore, large datasets of points (each generated point representing a specific system configuration) can be analysed to uncover trends when moving to larger or smaller system sizes and multiples.

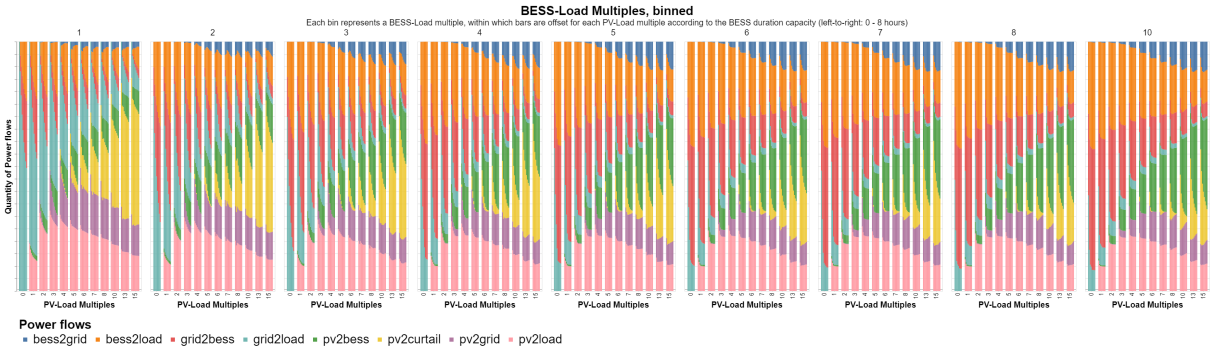

Figure 12: Normalised power flow distribution of for different size multiple of PV and BESS