Optimisation Code Operating Guidelines

This document explains how to use the LCOA optimisation model for single cases. It briefly explains how to install the code, and the relevant modules required for operation; it then provides a simple explanation of how to operate the code, and how results can be interpreted.

Code Installation

The code can be cloned from GitHub. Having cloned it, the user may then edit the 'back end' of the code, although this introduces the risk of error.

The code must be run in Python, and we recommend installation of Python 3.10, which should be kept up to date. Any Python IDE can be used; PyCharm (which is open source software) was used for development, but other options, such as Spyder, Jupyter or direct use of the terminal, are also suitable.

The following modules must be installed in order for the code to operate. The modules can be installed using pip or conda; we recommend using only one of these installation platforms and not both.

- pandas
- numpy
- pyomo
- pypsa

The most important dependency of the model is pypsa; if there are any difficulties with its installation, a useful explanation of installing both python and the software is available here: https://pypsa.readthedocs.io/en/latest/installation.html. It can also be cloned from the GitHub if preferred.

The model also requires an optimisation solver. The problem is a Mixed Integer Linear Program, for which a number of solvers are available. For basic usage, the free online solver 'glpk' is recommended, and can be installed by pip or conda installation. If the solver used is 'glpk', Pypsa will automatically output that ramping rate constraints are being rejected; however, the code has overwritten this requirement, as the user will be informed. Other free online solvers exist (e.g. cbc), but the code has not been tested using these solvers and may be very slow.

Note that when using glpk, the model can take a long time to solve (> 20 minutes). If faster solution is required, we recommend the use of Gurobi, which is a commercial solver requiring a license to be downloaded. Details for installation are available here: https://www.gurobi.com/downloads/

Code Operation

The user should run the code from within their preferred Python IDE by calling the 'main.py' file. The file will draw on the base plant design saved in the

'Basic_ammonia_plant' directory, which has cost and efficiency information. The code will then guide the user through a series of prompts that enable it to solve.

If additional renewable resources are desired, or the user wishes to remove a component, the simplest way to do so is to set a very high capital cost for that component, in which case the model will typically choose not to install it (In the default version, this is applied to SolarTracking and Grid). Alternatively, they can remove those components from the plant by editing the 'links.csv' file, the 'stores.csv' file, and the 'generators.csv' file.

The user also enters efficiency information for energy transfers between states in the links.csv file, all of which are measured in MWh. Flows of hydrogen are transformed between mass and energy using the higher heating value, which is 39.4 MWh/t (141.9 GJ/t); the same is done for ammonia, for which the HHV is 6.25 MWh/t (22.5 GJ/t).

PyPSA, which is the superstructure for the optimisation problem, requires that capital costs be paid in their entirety over the operating period. In this modelling, in order to enable rapid solution, we consider just a single operating year. Therefore, directly entering the installed capital cost of the plant equipment into PyPSA will not return accurate results (because a full capital investment is usually repaid over many operating years). The capital cost inputs must therefore be adjusted to reflect the representative operating period modelled here.

The user can do this themselves in the relevant .csv files in the Basic_ammonia_plant directory. A spreadsheet (Discount Rate Converter) is also included in the distribution in order to make estimating the Discount Rate easier. The code can annualize the costs for the user if they require, and will prompt them for relevant data with which to do so (although the discounting done by the code is very simple). The default information provided has already been annualized and should not be adjusted by the code.

The weather data associated with the specific location under consideration must also be provided. The code user needs to create this information in a separate input file. An example input file has been included, but any file (in the same format) is suitable, and the user will be prompted by the code for the name of the specific file they would like to use. It is important that one year of weather data is used to match the annualization of the capital costs, and the user will be warned if there appears to be a problem with the weather data.

A full profile of weather data must be provided for each generator listed in generators.csv; if there is a mismatch between the files, then the code will refuse to run. If the user does not wish to remove data associated with a type of energy generator from the generators.csv file, then they can set the entire operating profile to 0, in which case it will not be installed by the model.

In the default version of the code, the user should provide the name of the weather data file to the code on line 84, without a file extension (i.e. when the code is called). The user can also provide an extra string which will be appended to the name of the input data when the output spreadsheet is created. If the user would rather be guided through the process, they can leave the file_name and extension blank, in which case the code will prompt the user for that information.

Code Results

Results are output into an excel spreadsheet. The user can then plot either directly in excel or by using a python package like matplotlib.pyplot.

Note that if a component in the model has no associated the model will tend to allocate it a very large size.

Ammonia Quirks

Ammonia is a bit more complex to make than hydrogen, so a few additional adjustments are included:

i) HB plants are usually not considered to be entirely flexible, so back-up power is needed. The minimum rate of the HB is specified in the links.csv spreadsheet. Back-up energy can come from a battery or from a hydrogen fuel cell.

ii) The hydrogen fuel cell is simply modelled as a link between the hydrogen bus and the power bus with a target efficiency.

iii) The battery is a little more complex; it is modelled as having an interface (i.e. a maximum charging/discharging power) and a storage capacity. This is a reasonable approximation, but some error may be introduced from the decoupling of the power and energy storage components. The charging and discharging interfaces are modelled separately; the cost is associated with the charging interface, and the discharging interface is constrained to be the same size as the charging interface, adjusted for efficiency losses.

iv) The HB plant also cannot ramp very quickly. The ramp rates are specified in the links.csv spreadsheet, and implemented using additional pyomo constraints. Pypsa will warn that it cannot enforce ramp rate constraints; however, the new pyomo constraints imposed over the top of pypsa ensure the ramp limitations are held.

v) There are four constraints on the HB ramp rate: two hard constraints and two soft constraints (for hard and soft, there is a constraint on both upwards and downwards ramps). The hard constraints cannot be violated, and will keep the plant to within the rate specified in links.csv.

vi) The soft constraints are trickier. They are modelled as an entirely separate network of busses (RampPenalty to RampPenaltyDest), as a pseudo-power flow from a power source (RampDummy) via a link (PenaltyLink). I call it a pseudo-power because it's just easier to do the modelling in PyPSA by modelling the flow as a 'power', but it's really just a quantity with arbitrary units. The use of the PenaltyLink has a low operating cost (the specific value of the operating cost is arbitrary, but should be very small); the pseudo-power flow must be equal to the ramp rate of the ammonia plant. The operating cost of this adjacent network functionally imposes a very small penalty on the LCOA. This penalty is well within the margin of error of the LCOA estimation, and smooths out the results and prevents the model from predicting arbitrary ramping or 'zig-zagging'. If you don't understand what I

mean, try turning the operating cost of the PenaltyLink off – it won't change the headline results much, but if you plot the plant operation it's much less smooth.

vii) To impose all four constraints on the HB ramping, and the constraint on the battery charging and discharging interface, pyomo constraints are implemented. This means the model can only be solved using pyomo. It is possible to implement the constraints using linopt as well, but to avoid duplication I have opted for pyomo, which is simpler to understand. The model will call and implement those constraints.