Ecole Nationale des Ponts et Chaussées

Electrolysis and H2 Storage Modeling

Nicolas Bessin, Edgar Duc, Baptiste Vert

May 15, 2024

Contents

1 Overall presentation of the problem

This project, in collaboration with AirLiquide, aims to model a system of hydrogen production. The situation is as follows : we are an hydrogen producer, and we want to provide hydrogen at a constant rate (we suppose our client has a constant need of hydrogen, e.g. a steel-making factory). To produce this hydrogen, we want to use a dedicated renewable (wind & solar) energy park, and import energy from the grid / curtail or export energy as little as possible.

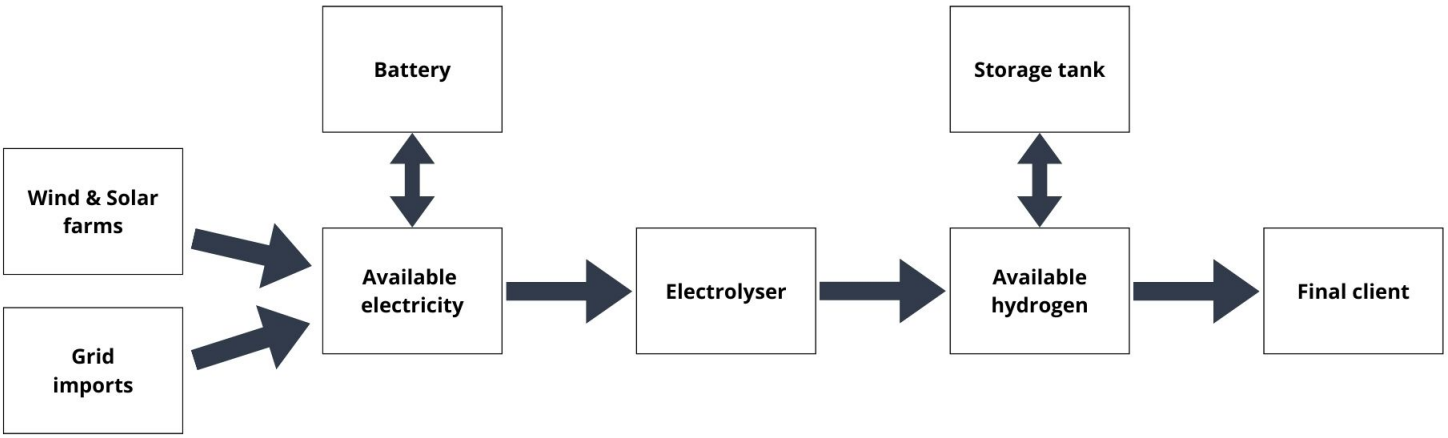


Figure 1: Illustration of the studied system

To be able to handle the inherent variations of the production capacities of our renewable electricity production, our system also has the following two means of storing energy :

- A battery on the electricity input size
- A hydrogen storage tank on the output size

However, because we can't know in advance the generation profiles of our wind & solar parks, we a priori don't know how to pilot our system (i.e., when to charge / discharge the battery, when to load / unload the tank, etc ...)
The main goal of the project is thus to find ways to pilot our hydrogen production system as best we can while only having generation profiles for periods at a time.
All the implementations of the methods presented in this report can be found on the Github repository [3].

2 Modeling the electrolyser - Implementation of a LP solver

First, we need to have a model / way to pilot the hourly behavior of the electrolyser when we know the generation profiles . We assume (for simplicty's sake) that the energy consumption is linear with respect to the production level. We also assume the energy coming from our dedicated win farm is free, and what we want to minimize is the imports / exports, and the installation costs when sizing the system. This leads us to the following LP formulation :

2.1 Parameters & Variables

Parameters

D : Customer demand.
$W^P \in [0,1]^T$: Generational capacity profile for wind.
$S^P \in [0,1]^T$: Generational capacity profile for solar.
$P_{\text{grid}} \in \mathbb{R}_+$: Price to import electricity from the grid (in €/MWh).
$P_{\text{curt}} \in \mathbb{R}_+$: Price to curtail electricity (in €/MWh).
$P_{\text{change}} \in \mathbb{R}_+$: Price to change the production of hydrogen (in €/kg of change).
$E_{\text{bat}} \in \mathbb{R}_+$: Efficiency of the battery (discharge rate, per hour).
$F_{\text{bat}} \in \mathbb{R}_+$: Maximum flow of electricity to / from the battery (in MW).
$E_{\text{elec}} \in \mathbb{R}_+$: Efficiency of the electrolyser (in MWh/kg).
$C_{\text{elecc}} \in \mathbb{R}_+$: Cost to build the electrolyser (in €/MW of capacity).
$C_{\text{bat}} \in \mathbb{R}_+$: Cost to build the battery (in €/MWh of capacity).
$C_{\text{tank}} \in \mathbb{R}_+$: Cost to build the hydrogen tank (in €/kg of capacity).
$C_{\text{wind}} \in \mathbb{R}_+$: Cost to build wind generation (in €/MW of capacity).
$C_{\text{sol}} \in \mathbb{R}_+$: Cost to build solar generation (in €/MW of capacity).
$c_i \in \mathbb{R}_+$: Initial charge of the battery (in MWh) (0 by default).
$s_i \in \mathbb{R}_+$: Initial stock of hydrogen (in kg) (0 by default).
$c_f \in \mathbb{R}_+$: Final charge of the battery (in MWh) (Not constrained by default).
$s_f \in \mathbb{R}_+$: Final stock of hydrogen (in kg) (Not constrained by default).

Parameters or variables, depending on the chosen model

$W^C \in \mathbb{R}_+$: Capacity of the wind generation (in MW).
$S^C \in \mathbb{R}_+$: Capacity of the solar generation (in MW).
$E^C \in \mathbb{R}_+$: Capacity of the electrolyser (in MW).
$B^C \in \mathbb{R}_+$: Capacity of the battery (in MWh).
$T^C \in \mathbb{R}_+$: Capacity of the hydrogen tank (in kg).

Variables

$p \in \mathbb{R}_+^T$: Current production level of hydrogen (in kg).
$c \in \mathbb{R}_+^{T+1}$: Current charge of the battery (in MWh).
$s \in \mathbb{R}_+^{T+1}$: Current stock of hydrogen (in kg).
$e^{\text{grid}} \in \mathbb{R}_+^T$: Electricity imported from the grid (in MWh).
$e_{\text{curt}} \in \mathbb{R}_+^T$: Electricity curtailed (in MWh).
$f^{\text{bat}} \in \mathbb{R}_+^T$: Electricity to / from the battery (in MWh).
$f^{\text{tank}} \in \mathbb{R}_+^T$: Flow of hydrogen to / from the tank (in kg).

$p^{\text{change}} \in \mathbb{R}_+^{T-1}$: Absolute change in hydrogen production (in €).

2.2 Constraints

1. Initial and final conditions:

$$\begin{aligned} \mathrm{c}_1 &= \mathrm{c}_i \\ \mathrm{s}_1 &= \mathrm{s}_i \\ \mathrm{c}_{T+1} &= \mathrm{c}_f \\ \mathrm{s}_{T+1} &= \mathrm{s}_f \end{aligned} \tag{1}$$

2. Electricity balance:

$$\forall t \in [\![1;T]\!], S_t^P \times S^C + W_t^P \times W^C + e_t^{\text{grid}} - e_t^{\text{curt}} = E_{\text{elec}} \times p_t + f_t^{\text{bat}} \tag{2}$$

3. Demand satisfaction:

$$\forall t \in [\![1;T]\!], p_t = D + f_t^{\text{tank}} \tag{3}$$

4. Battery flow :

$$\forall t \in [\![1;T]\!], E_{\text{bat}} \times c_t + f_t^{\text{bat}} = c_{t+1} \tag{4}$$

5. Tank flow :

$$\forall t \in [\![1;T]\!], s_t + f_t^{\text{tank}} = s_{t+1} \tag{5}$$

6. Flow of electricity / hydrogen :

$$\forall t \in [\![1;T]\!], -F^{\text{bat}} \leq f_t^{\text{bat}} \leq F^{\text{bat}} \tag{6}$$

7. Electrolyser capacity:

$$\forall t \in [\![1;T]\!], p_t \times E_{\text{elec}} \leq E^C \tag{7}$$

8. Battery & tank capacity:

$$\begin{aligned} \forall t \in [\![1;T]\!], & 0 \leq c_t \leq B^C \\ \forall t \in [\![1;T]\!], & 0 \leq s_t \leq T^C \end{aligned} \tag{8}$$

9. Production Change penalty:

$$\begin{aligned} \forall t \in [\![2;T]\!], & p_t - p_{t-1} \leq p_t^{\text{change}} \\ \forall t \in [\![2;T]\!], & p_{t-1} - p_t \leq p_t^{\text{change}} \end{aligned} \tag{9}$$

2.3 Objective

Minimize the overall cost:

$$\begin{aligned} \min_{\text{Variables}} & P_{\text{grid}} \sum_{t=1}^{T} e_t^{\text{grid}} + P_{\text{curt}} \sum_{t=1}^{T} e_t^{\text{curt}} + P_{\text{change}} \sum_{t=1}^{T-1} p_t^{\text{change}} \\ & + W^C \times C_{\text{wind}} + S^C \times C_{\text{sol}} + E^C \times C_{\text{elec}} + B^C \times C_{\text{bat}} + T^C \times C_{\text{tank}} \end{aligned} \tag{10}$$

Note : If we give the different capacities as fixed parameters instead of variables, the corresponding costs are constants in the final objective, but we can keep them in to have a generalized formulation of the problem.

2.4 Parameters values

We will use the following values when calling the solver for the remainder of this report, unless specified otherwise.

For the basic parameters, we use the values given in the project description.

E_{bat}	F_{bat}	E_{elec}
10% (loss / month)	100 MW	0.05 MWh / kg

Here we use high values for the costs, to force the solver to use the dedicated park as much as possible.

P_{grid}	P_{curt}	P_{change}
1000 €/ MWh	500 €/ MWh	10 €/ kg

Finally, we use the following values for the costs of the different capacities.
For the electrolyser, battery and tank, we use the values given in the project description.
For the wind and solar capacities, we use the values from [1]

C_{elec}	C_{bat}	C_{tank}	C_{wind}	C_{sol}
480 €/ MW	50 €/ MWh	437 €/ kg	110600 €/ MW	84650 €/ MW

We implemented this LP problem using the JuMP modeling framework [4] for the Julia programming language.

3 Size calculations

3.1 A note on the data we use

In the remainder of this report, we will use data from Open Power system data [2], mainly the hourly generation profiles for solar & wind in Germany in the years 2014 to 2018. The profile value, in $[|0, 1|]$, is the share of the installed capacity that is actually producing. For example, if the installed wind capacity is 100 MW, and the wind profile for the current hour is 0.6, it means the real output is 60 MW.

3.2 Determining the best size

Before trying to optimally pilot the system, we have to know the wind and solar capacities, as well as the battery, tank and electrolyser capacities we are going to build. We can use the solver to compute those capacities, by not giving them as fixed parameters, but as variables to optimize. To integrate those variables into the cost, we use their cost per unit of capacity, over the time horizon. We also further restrict the battery capacity to cover at most 6 hours of productions, to keep a realistic battery capacity.

Here we can use the full 5 years of data, to have the best possible estimation of the optimal capacities.

3.3 Size calculations results

We get the following installed capacities. In the remainder of this report, those are the capacities will will use for the different components of our system.

Battery	Tank	Electrolyser	Solar	Wind
300 MWh	50978 Kg	2143 Kg / h	196 MW	132 MW

Table 1: Capacities of the system's components

Further, plotting the hourly battery charge level & hourly tank stock level, we see the desired effet of short term storage, with a high frequency of fluctuations for the battery, as opposed to a more long-term storage effect with the tank, and lower fluctuation frequency.

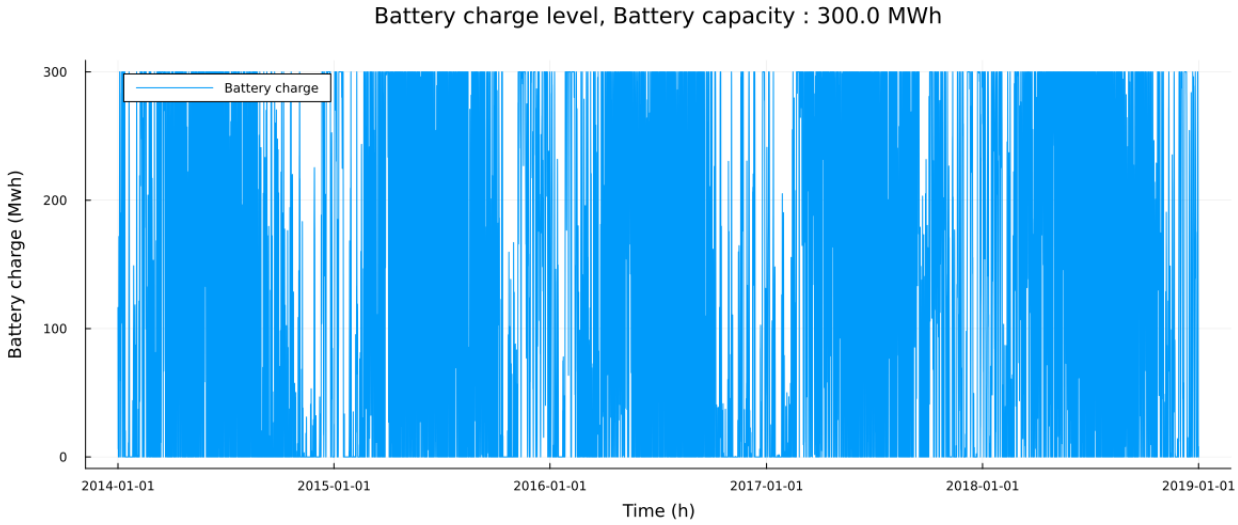


Figure 2: Hourly battery charge level over the full period

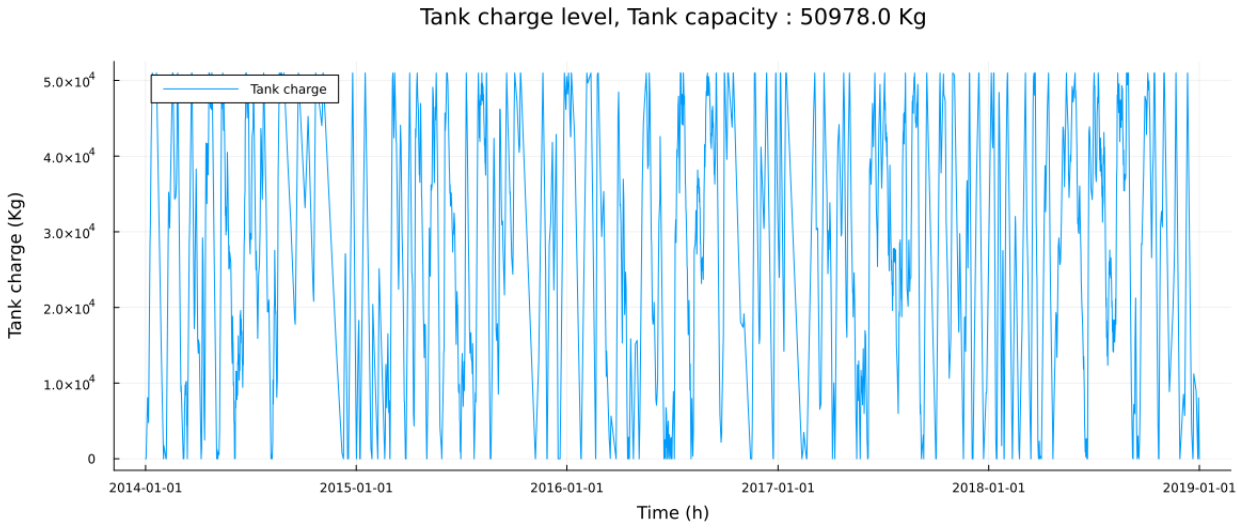


Figure 3: Hourly tank stock level over the full period

4 First approach : Dynamic Programming

As discussed in the introduction, we can't use the previous LP to get the behavior of the system, because we don't know in advance the generation profiles. One way to alleviate this is to have a long-term policy to pilot the hydrogen stock :

- Suppose we can have accurate predictions of the generation profiles one week in advance
- Pilot the electrolyser optimally to reach a constrained level of stock at the end of the current week, given by the policy
- Repeat until the end of the long-term period (e.g. a period of 52 weeks)

This way, by only supposing we can predict the profiles one week in advance, we can study what policy gives us the best results, as in, the closest to the optimal we could have computed if we knew the whole profiles in advance.

Note : it could more realistically be one day in advance, but we will keep one week for the sake of computation times

One way to compute such a policy is to use dynamic programming.

4.1 Bellman equation

Since we supposed we know the generation profiles for the coming week, and because we saw previously that the battery has only a very short term effect, we will write our system state as follows :

$$s_t = (x_t^s, c)$$

- x_t^s is the current stock in the tank,
- c designates a "class" for the generation profile of the current week (i.e. an indicator of the available energy quantity)
- To remove complexity, we use a discrete state space of N_{levels} values, where x_t^s take values by e.g. 1/10th of the tank capacity (corresponding to $N_{\text{levels}} = 11$)
- The action at time t is the choice of stock level we want to reach at time $t+1$
- The cost function is given by solving the LP problem over the period $[t, t+1]$, using several possible profiles to get an expected value
- For the battery, we simply impose continuity of the charge level

The dynamic programming equation is given by:

$$V_T(x_T^s, c) = 0$$

$$V_t(x_t^s, c_t) = \min_{x_{t+1}^s} \left\{ \mathbb{E}_{W^P, S^P, c_{t+1}} [C(x_t^s, c_t, x_{t+1}^s) + V_{t+1}(x_{t+1}^s, c_{t+1})] \right\}$$

where $C(x_t^s, c_t, x_{t+1}^s)$ is the cost of reaching x_{t+1}^s from x_t^s over the period $[t, t+1]$, if the generation profile is of class c_t.

Note : Every invocation of the LP is now done with the fixed capacities we computed previosuly.

4.2 Profiles classification

We now split our data into a list of week-long profiles, and we further split this list of profiles into training / validation profiles (e.g. we keep the weeks from 2018 as validation, and use the rest as training)

Using the previously defined notations, the hourly available energy in the dedicated park is :

$$E = W^C \times W^P + S^C \times S^P \in \mathbb{R}_+^T$$

. The total available energy over a full week is thus :

$$\sum_{t=1}^{T} E_t := \sum E$$

Let's denote $\mathcal{E}$ the list of available energy profiles in the training data. One way easy way to classify the profiles into k classes is :

- Compute the min and max energy available : $e_{\min} = \min_{E \in \mathcal{E}}(\sum E)$, $e_{\max} = \max_{E \in \mathcal{E}}(\sum E)$
- Compute $k+1$ uniformly distributed thresholds e_i between $e_{\min}$ and $e_{\max}$
- Assign the classes as follows : $c = 1$ if $\sum E \in [e_1 = e_{\min}, e_2]$, ..., $c = k$ if $\sum E \in [e_k, e_{k+1}]$

We also use the previously computed thresholds (computed on the training data) to assign classes to the validation profiles. By counting the number of profiles in each class in the training data, we get the "probabilities" of each class.

4.3 Computing the expected value

To compute the expected value, we use plausible profiles for each state (x^s, c). Let's denote by N_{ev} the number of profiles we use to compute the expected value. There are several ways to get those plausible profiles, but the simplest is to just pick a full week of class c from the training data.

Further, since we only know the class of the week t, not that of week $t+1$, we need to use the probabilities of each class we computed previously.

The transition function for a given profile is given by solving the LP problem with this profile.

Denoting by W_i^P, S_i^P the i th profile picked for the computation (order does not matter), and by p_c the probability of class c, our computation becomes :

$$\mathbb{E}_{W^P, S^P, c_{t+1}}[C(x_t^s, c_t, x_{t+1}^s) + V_{t+1}(x_{t+1}^s, c_{t+1})] = \sum_{i=1}^{N_{ev}} \frac{1}{N_{ev}} \sum_{c=1}^{k} p_c[\text{LPCost}(W_i^P, S_i^P, x_t^s, x_{t+1}^s) + V_{t+1}(x_{t+1}^s, c)]$$

Where $\text{LPCost}(W_i^P, S_i^P, x_t^s, x_{t+1}^s)$ is the operating cost of the system for going from $x^s - t$ to x_{t+1}^s with the given profiles. Note that we can consider this cost because we consider that the imports form the grid / curtailment are unlimited (even though costly), so almost every transition is feasible. The only non feasible transition are if we cannot fill or empty the tank fast enough (e.g if we considered periods that are 24 hours long, we can't empty the tank over a single period, since a full tank represents around 51 hours of demand). We don't have to worry about this phenomenon for periods of 1 week, every transition is feasible.

4.4 Simulating a policy

Having computed a policy using the previously described method, we need a way to simulate its effect. Let's denote N_p the number of periods over which we computed the policy (e.g. $N_p = 52$ to simulate a whole year). We now denote by $\mathcal{W}$ and $\mathcal{S}$ the lists of validation profiles (in order, i.e $\mathcal{W}[1]$ is the wind profile for the first week of the validation period). Further, we denote by A the policy. As we saw previously, $A[t, x, c]$ gives us the action we must take when starting the week t with stock x, and if the week is "of class" c

Algorithm 1 Policy simulator

Require: $N_p, \mathcal{W}, \mathcal{S}, A$
  cost $\leftarrow 0$
  stock $\leftarrow s_0$
  for $i \in [|1, N_p|]$ **do**
    $E \leftarrow \mathcal{W}[i] \times W^C + \mathcal{S}[i] \times S^C$
    $c \leftarrow \text{class}(E)$
    $a \leftarrow A[i, \text{stock}, c]$
    cost $\leftarrow$ cost $+ \text{LPCost}(\mathcal{W}[i], \mathcal{S}[i], \text{stock}, a)$
    stock $\leftarrow a$
  end for

This algorithm gives us the cost of applying the policy A over the validation period (still using the assumption at at the beginning of every sub-period of one week, we exactly know the weather for the week to come, i.e. that the production plan given by the LP Solver is exact and optimal.)

To get an idea of the value of the policy, simply knowing the cost it returns is not enough, since it heavily depends on the parameters we chose previously. We need to compute a relative gap to the optimal solution. Thus we compute the optimal solution as follows : The optimal way to pilot the system is the production plan given by the LP solver if we had known the profile for the whole validation period in advance, and not just week by week. We then compute the relative gap between the two :

$$\text{gap} = \frac{\text{cost}_{\text{policy}} - \text{cost}_{\text{optimal}}}{\text{cost}_{\text{optimal}}} \geq 0$$

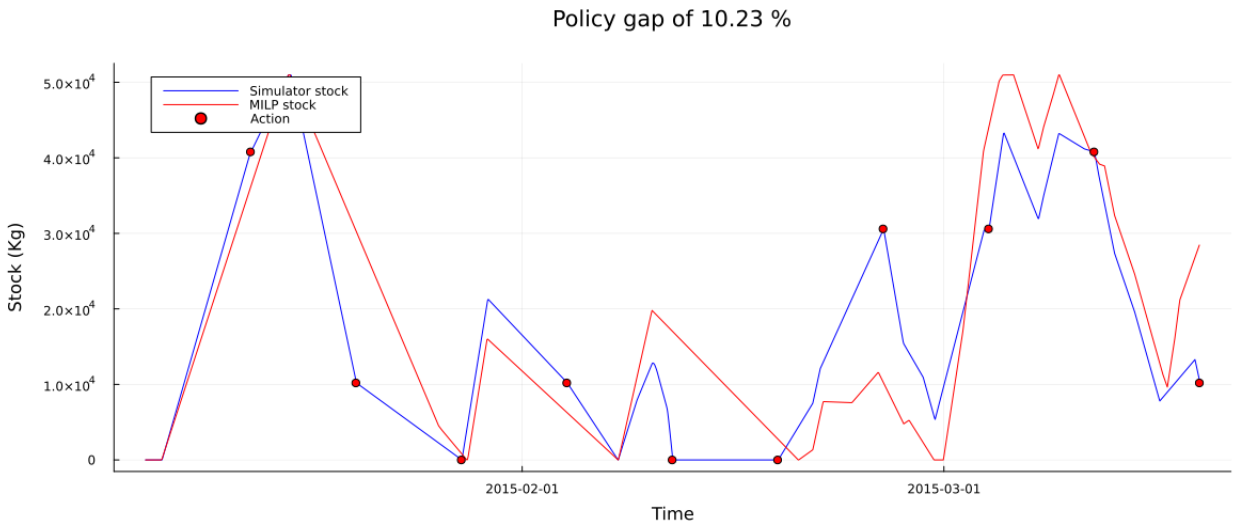


Figure 4: Simulating the policy over 10 weeks, $N_{ev} = 5$, $k = 5$, $N_{\text{levels}} = 6$

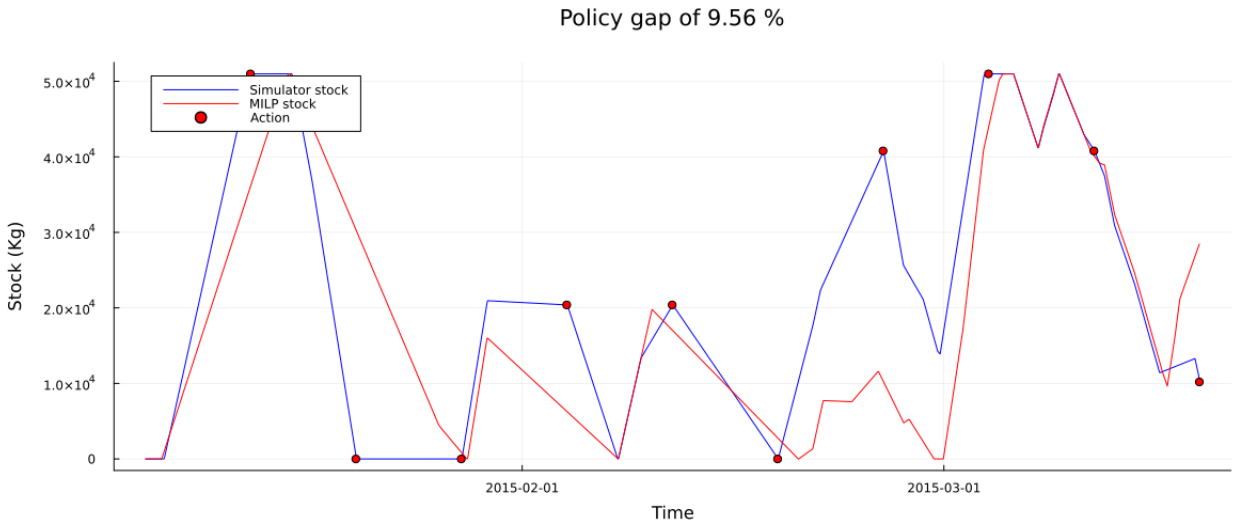


Figure 5: Simulating the policy over 10 weeks, $N_{ev} = 5$, $k = 10$, $N_{\text{levels}} = 6$

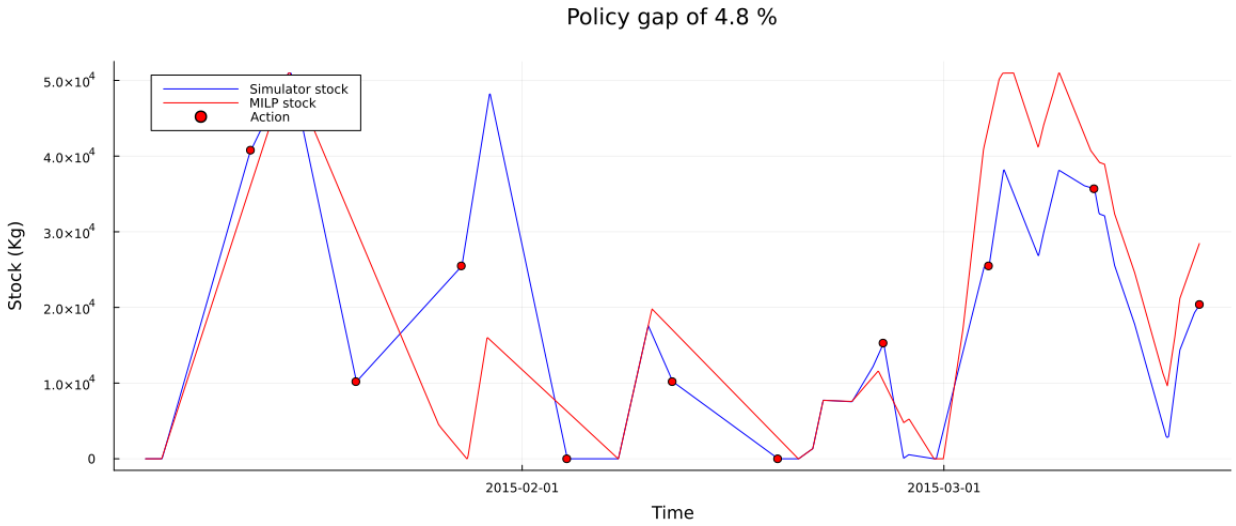


Figure 6: Simulating the policy over 10 weeks, $N_{ev} = 5$, $k = 5$, $N_{\text{levels}} = 11$

In those three simple example, we plot the stock level over time for both the policy simulation (blue) and the LP solver (red). The red dots correspond to the action taken at the beginning of every week, and thus at the end of every week, the policy stock level is exactly at the action taken.
In all of those example, we see that our policy gives us a good approximation of the optimal (as in, the stock level curve roughly has the same shape as the optimal curve).
Further, looking at figures 4 and 6, we see that the gap is lower when using a higher precision on the stock level, which is something we expect (and especially since $\{0, \frac{1}{5}, .., 1\} \subset \{0, \frac{1}{10}, ..., 1\}$, a policy with $N_{\text{levels}} = 6$ is also a valid policy with $N_{\text{levels}} = 11$, thus we must have that the cost with the higher level precision is better).
Comparing figures 4 and 5, we also see that increasing the precision on the classification of the profiles also gives us better results (but not to the same extent as with the state levels)

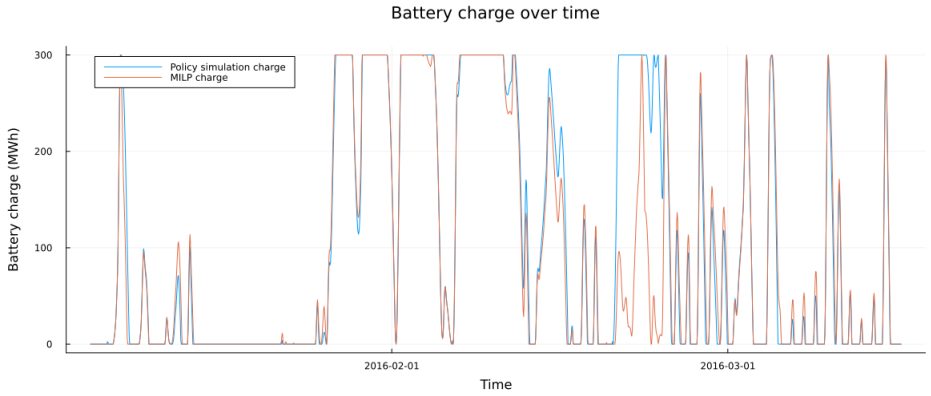


Figure 7: Hourly battery levels over 10 weeks, $N_{ev} = 5$, $k = 5$, $N_{\text{levels}} = 11$

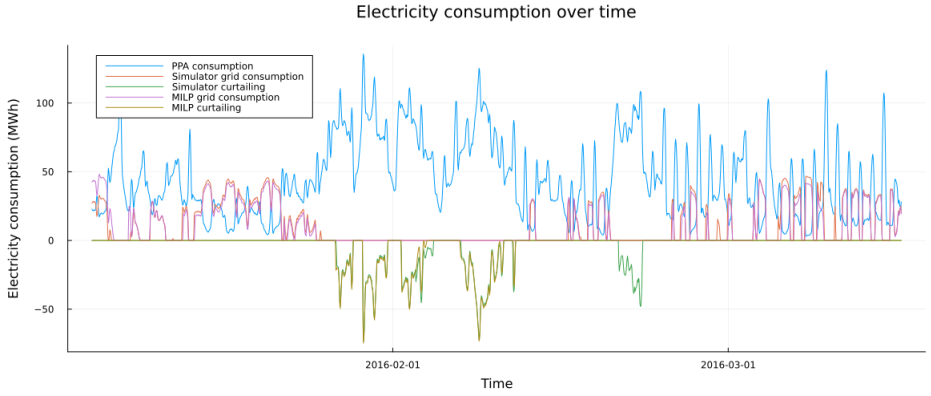


Figure 8: Hourly electricity consumption over 10 weeks, $N_{ev} = 5$, $k = 5$, $N_{\text{levels}} = 11$

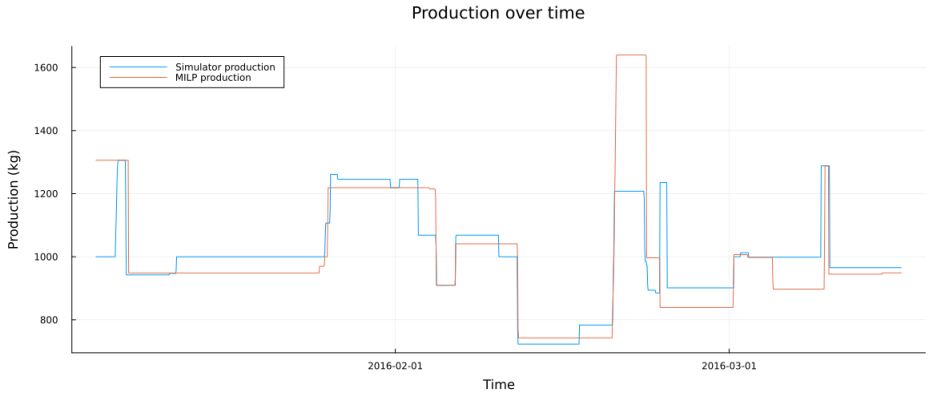


Figure 9: Hourly production level over 10 weeks, $N_{ev} = 5$, $k = 5$, $N_{\text{levels}} = 11$

Figures 7, 8, 9 compare the hourly battery levels, electricity consumption (from dedicated park, grid and curtailing) and the production levels between the optimal solution obtained with the full LP, and the simulated policy. We see that the metrics we get with the policy are close to the optimal metrics, indicating that the policy gives us a good approach to pilot our system.

Still, since the computation time for the policy is around $\mathcal{O}(N_p N_{\text{levels}}^2 k N_{ev})$, increasing the precision, while bringing better results, is done at the cost of increasing the computation time for the policy

4.5 Tuning the parameters

Using a rough cross validation (testing the same parameters using different years as the validation data), we find the following parameters give the best compromise of computation time / results

Number of stock levels	Number of classes	Number of profiles used
11	5	10

Table 2: Best parameters / sub routine for solving the dynamic programming equation

Using those parameters, and computing then simulating the policy over a full year (taking approximately 50 minutes to compute, hence the lower amount of parameters tested and shown), we get :

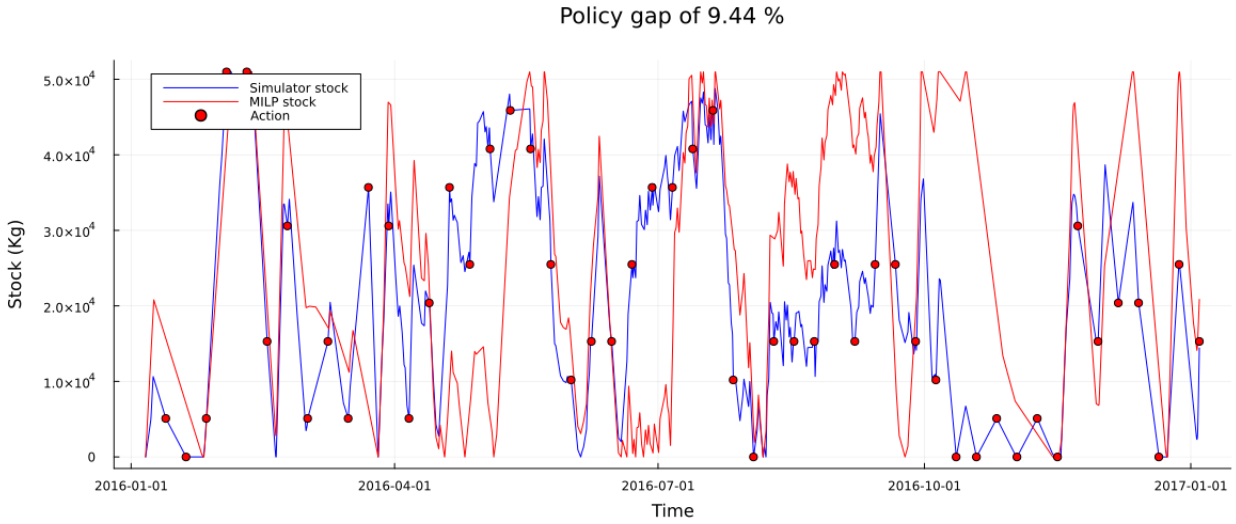


Figure 10: Simulating the policy over 52 weeks, $N_{ev} = 10$, $k = 5$, $N_{\text{levels}} = 11$

As we saw previously, the policy we compute using the Bellman equation gives a relatively good approximation of the optimal piloting we could have computed if we knew the full profiles in advance.

Second approach with the heuristic and the statistical concepts

This part will be split into different sub parts. The first part will present the choices of modeling made for the study of the impact of the final constrained levels of hydrogen on the real cost. The second part will develop the modeling of the function of transposition of the cost, whose aim is to give the real cost from a production plan based on predicted data. The third part will present the subroutine used by the heuristic used in this modeling, and the last part will present the final regression function, which can be seen as the aim of this project through this modeling based on a heuristic and on some statistical concepts.

5 Study of the impact of the final constrained level of stock

5.1 Study of the methods

First, here we will use a default model for prediction.

We fix the number of weeks Nw over which we build our optimal production plan : we choose a first true week, then we predict Nw-1 weeks, we also store the true Nw-1 weeks. Then we run our optimization program in the two cases, the first one corresponds to the case with no constraints for the final levels of hydrogen and the second is the opposite, and we compare the two optimal costs obtained by running the optimization program.

After fixing the time period, for example we want to start at the 37th week work until the 43th week. We need to choose the true year, the one from which only the first week will be extracted, then we need to complete the 5 following weeks, to so there are different possibilities, either we used the 38th, 39th,..., 43th weeks from other years picked randomly, or we can use a model that will have the first week as an input and will be able to predict the 5 following weeks.

6 The problem of transposition of the production plans

6.1 Explanation of the problem

Actually we have a problem of transposition between our true situation and our predicted situation. Indeed, the comparison of the two optimal costs does not really make sense, since we are in two different situations. The idea would be to use our predicted weeks to get a production plan where we actually constrained some final levels of hydrogen, then we transpose this production plan into the real situation.

TO be more precise, the issue is that our predicted production plan may not be feasible in the real situation, for example if we planned to used 70MW from the dedicated park at 12 o'clock on a Saturday, but there are only 40MW which are available, then we need to review our production plan, but the question is how much we have to review this plan, could we just not only use 30MW more at the following hour? But if it is not possible, can we do it at the second following hour? But if we had been so bad at predicting and this consequently resulted in being unable to fulfill the demand, then we have a problem. We could imagine tremendous cost to punish the incapacity of fulfill the demand instead of taking it as a constraint. So the point is that we need to create a transposition model, with costs for changes and not fulfilling the demand, these would be mistake-costs.

So to deal with this issue about transposition, we will just use the energy from the grid, which is definitely an easier way to deal with these false predictions. Indeed, we suppose here that the grid has infinite capacities, the only problem is that its electricity is costlier than the one from the park. So if we predicted and used more electricity from the park that was actually available, we complete with electricity from the grid which will increase our final cost. Therefore we can transform an unfeasible production plan into a feasible one.

6.2 The function of the transposition of the cost

Here we develop how this function of transposition is designed.

$\mathbf{N_w} \in \mathbb{N}^* \setminus \{1\}$: number of weeks over which the predicted production plan is computed

$\mathbf{x_s} \in \mathbb{N}^{N_w - 1}$: vector of the constrained final levels of stored hydrogen.

Opt_park$(x_s) \in \mathbb{R}_+^T$: represents the quantity of electricity from the **dedicated park** used at each hour according to the predicted production plan which depends on x_s.

Opt_Ener_Market$(x_s) \in \mathbb{R}_+^T$: represents the quantity of electricity from the **grid** used at each hour according to the predicted production plan which depends on x_s.

Real_park $\in \mathbb{R}_+^T$: represents the quantity of electricity generated by the park at each hour in the real case.

Real_price $\in \mathbb{R}^T$: represents the prices on the grid at each hour in the real case, here prices can be negative as discussed previously.

$$\begin{aligned} F\big(\text{Real\_park, Real\_price,Opt\_park}(x_s)\text{,Opt\_Ener\_Market}(x_s)\big) &= \sum_{t=1}^{T}\big[\text{Opt\_park}(x_s)_t - \text{Real\_park}(x_s)_t\big]^+\text{Real\_Price}_t \\ &+ \sum_{t=1}^{T}\big[\text{Opt\_park}(x_s)_t - \big[\text{Real\_park}_t - \text{Opt\_Ener\_Market}(x_s)_t\big]^+\big]^+\text{Real\_Price}_t \\ &= \sum_{t=1}^{T}\big[\text{Opt\_park}(x_s)_t + \text{Opt\_Ener\_Market}(x_s)_t - \text{Real\_park}_t\big]^+\text{Real\_Price}_t \end{aligned} \tag{11}$$

6.3 Pseudo code for the algorithms

Our study is split into three parts :

1) The generation of predicted data: we fix the time period and then we generate prediction for the available energy from the dedicated park and for the prices on the grid.

2) The creation of the optimization problem with the predictions: this is here that all the modeling takes part with the constraints and the chosen objective functions which is composed of different costs (here we only consider the costs of energy from the grid as explained before).

3) The use of a heuristic : this is where the function of the transposition of the cost takes part, the aim will be to find the lowest real cost that could be obtained with different values for the constraints of the final levels of stored hydrogen.

Creation of the optimization problem with constrained levels of stored hydrogen

Here we have a slightly different optimization program that the one presented in the first part :

First, we need to add the constraints for the final level of stored hydrogen of the first $N_w - 1$ weeks. Secondly, we keep only the cost which corresponds to the consumption of electricity from the grid. Finally, we add the notion of efficiency for the storage of electricity in the battery, that means there is still hourly dissipation but there is also a loss of electricity during the transfer of the flow of electricity from the battery.

Parameters

Pred_park $\in \mathbb{R}_+^{N_h}$: predicted quantity of electricity generated by the dedicated park per hour in MW.
Pred_Price $\in \mathbb{R}_+^{N_h}$: predicted prices on the grid per hour in €/MW.

hyd_stored_level(also noted $\mathbf{x_s}$) $\in \mathbb{N}^{N_w-1}$: corresponds to the value of the minimal stock of hydrogen to have at the end of the first $N_w - 1$ weeks (in kg).

Objective function of the sub-optimization programm

$$f(\text{Ener\_Market}) = \sum_{t=1}^{N_h} \text{Pred\_Price}_t \text{Ener\_Market}_t \tag{12}$$

With this construction of the optimization problem, we can finally create our subroutine for the heuristic.

Algorithm 2 Subroutine of the heuristic

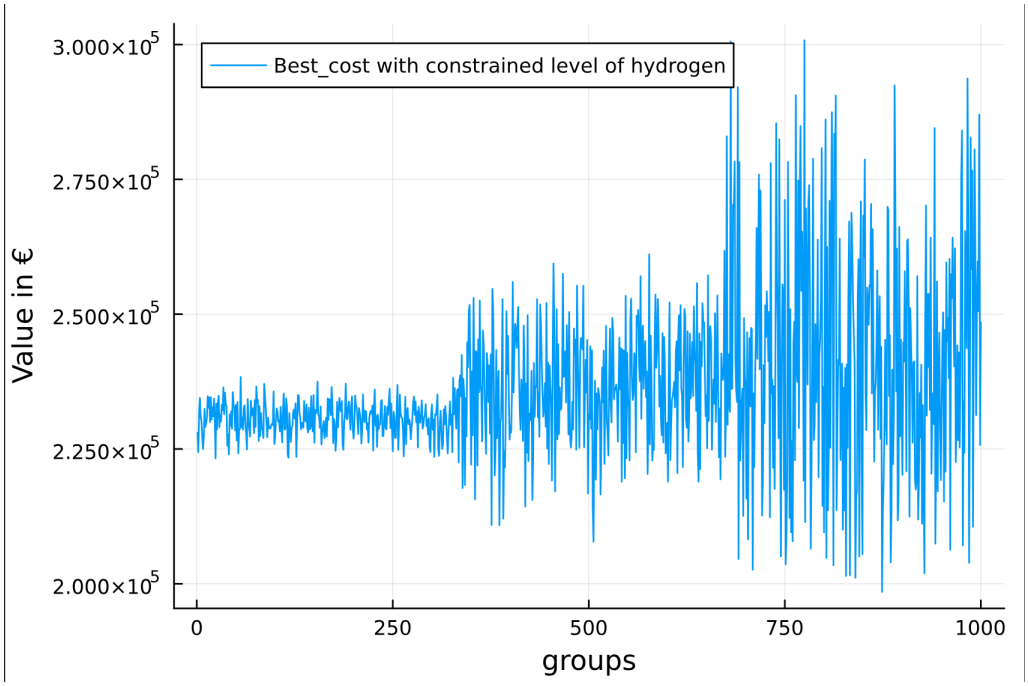
Require: Parameters of the precedent optimization problem (particularly x_s), Pred_park, Pred_price, Real_park, Real_Price
Ensure: Real_Cost
1: Opt_Ener_park(x_s) $\leftarrow$ Ener_park(x_s) ▷ solution of the optimization problem using all the inputs
2: Opt_Ener_Market(x_s) $\leftarrow$ Ener_Market(x_s) ▷ same origin
3: Real_Cost $\leftarrow F($Real_park, Real_price,Opt_park(x_s),Opt_Ener_Market(x_s)$)$ ▷ With F for the transposition of the cost

The third part with the heuristic will be described in the next section, but now we test our subroutine with random constrained final levels of stored hydrogen, to have a first look on how the real objective function, that is the function of transposition of the cost, behaves.

6.4 First look at the generation of random real costs

First we look at the optimal cost with random-generated constrained final level of hydrogen, it seems that the different optimal costs do not change a lot, like maybe one percent of difference. This would mean that the different stocks of hydrogen do not have a large impact on the cost. This is actually due do the fact that the constrained levels of hydrogen were too low compared to the optimal case where we did not have these constraints, that means these constraints are already verified in the optimal case. Indeed we can notice that the optimizer tends to saturate the capacities of storage around the middle of the period, and then it lets it decrease until the end. This is a first observation, so not necessarily true for all cases, it actually means that the range of values that the constrained levels of hydrogen can take needs to be large. So instead of restricting the random levels to be between 0 and 30% of the maximum capacity of storage of the tank, we let it go until 100% while generating more simulations.

Here is what we obtain with three different restrictions for the final constrained levels of hydrogen :



We observe through this first graphic that some evaluations with non-zero constrained levels have a lower value than the first evaluation that corresponds to the zero case. So with these predicted data constraining the levels has a positive impact on the diminution of the transposed cost, which is a good sign for the interest of the stock constraint method.

7 Study of the heuristic

7.1 Settings for the tests of the simulated annealing

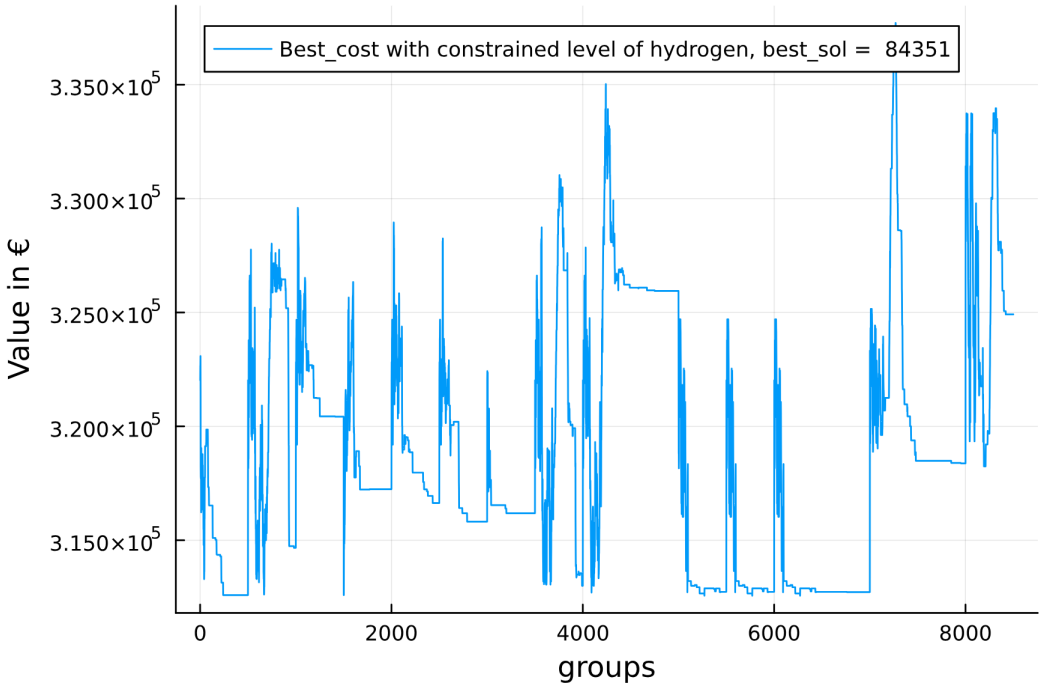
First as explained previously, we need to fix a date, to predict the available energies and prices per hour, then we need to first evaluate the precision of our predictions (since we have the real data), then we need to launch our heuristic with our set of parameters and our predicted data, and see which constrained levels of hydrogen will allow to get to the lowest real cost ,as fast as possible, with our predicted data.

So here the precision of our predictions can have a real impact on the behaviour of our algorithm. Therefor we will have to use better prediction that randomly picking weeks among years.

We also want to ensure reproducibility with our tests, indeed to study how our heuristic works it is better to fix our prediction and change its parameters at first, then obviously we will test our heuristic with other predictions, since in reality we will have different weeks with therefore different prediction, and so the errors will not be at the same hours.

To assure this reproducibility of the tests we will use the seed in Julia, which allows to generate random number while keeping a trace on them.

Here is a graphic showing iterations with the heuristic and different sets of parameters :



Parameters and variables of the heuristic

$\boldsymbol{\alpha \in (0,1)}$: the decreasing coefficient.

$\boldsymbol{S_p} \in \mathbb{N}$: the size of a patch

$\boldsymbol{\theta \in \{0,1\}}$: with $\boldsymbol{\theta = 0}$ if the last time split is constrained and $\boldsymbol{\theta = 1}$ otherwise.

$\boldsymbol{H\_ini\_lev} \in \mathbb{N}^{\boldsymbol{N_w}}$: initial solution

$\boldsymbol{T\_ini} \in \mathbb{R}^+$: initial temperature

Var : variance of the Gaussian law used to model the notion of neighborhood.

Here is the pseudo-code that allows to generate random solution in the Gaussian neighborhood of a previous solution, and the pseudo-code for the Metropois Criterion:

Algorithm 3 Gaussian neighborhood

Require: H_lev, Var
Ensure: New_H_lev
$r \leftarrow$ randint$(1, N_w)$ ▷ a random integer $\in [\![1; N_w]\!]$
r_list $\leftarrow$ randint$(1, N_w, r)$ ▷ a list of r random integers $\in [\![1; N_w]\!]$, without replacement
for i in r_list **do**
H_lev[i] = $\lfloor$randnorm$(H_lev[i], \text{Var})\rfloor$ ▷ Gaussian integer centered at the previous value
end for
New_H_lev = H_lev

Algorithm 4 Simulated annealing

Require: T_ini, S_p, θ, H_ini_lev, α, N_{iter}, Var, P_{ar}, Pred_park, Pred_Price, Real_park, Real_Price

Ensure: H_opt $\in \mathbb{N}^{N_w}$

Current_cost $\leftarrow$ Algorithm_1(P_{ar}, H_ini, Pred_park, Pred_Price, Real_park, Real_Price) ▷ here H_ini is equivalent to the previous x_s
Current_H_lev $\leftarrow$ H_ini
for i in 1:N_{iter} **do**
    for k in 1:S_p **do**
        H_lev $\leftarrow$ Algorithm_2(Current_H_lev,Var)
        new_cost $\leftarrow$ Algorithm_2(P_{ar}, H_lev, Pred_park, Pred_Price, Real_park, Real_Price)
        $\Delta \leftarrow$ new_cost - Current_cost
        if Metropolis_Criterion(T,Δ) **then**
            current_cost $\leftarrow$ new_cost
            Current_H_lev $\leftarrow$ H_lev
        end if
    end for
    T $\leftarrow (1-\alpha)T$
end for
H_opt $\leftarrow$ Current_H_lev

8 Project regression function

8.1 Setting of the regression context

This final part allows to finalize the object of the project developed in the second part, namely the impact of constraining the final levels of hydrogen to get the lowest transposed cost.

The context is, we have a fixed period, let us say it is the last day of the year, we want to elaborate an optimal production plan for the next 5 weeks. To do so, we have the parameters of the infrastructure such as the capacities, the efficiencies and the dissipation coefficient of the battery. In this study we consider that these parameters are fixed, so they are not taken into account in the regression problem, contrary to the data used for the prediction of solar and wind profiles and the prices of electricity on the grid. Indeed, since they are times series, each day we can add new data about solar and wind profiles and prices of electricity on the grid.

These parameters and data, associated with the fixed period, namely the date of the beginning of the production plan and the total number of time splits (here we chose weeks) over which we want to construct this production plan, represent the inputs of the regression problem.

The response variable is the vector of the values of the constraints for the final levels of stored hydrogen over the fixed time period (here the last time-split level can be constrained, but in the previous part we did not study that case).

So here we give the regression relation verified by the regression function denoted by $\boldsymbol{f_{D_a,P_r,N_T,\theta,N_{\text{res}}}}$:

$$\boldsymbol{Y = f_{D_a,P_r,N_T,\theta,N_{\text{res}}}(D_{a\_\text{new}\_\text{res}}(F_P))} \tag{13}$$

The efficiency of the second regression function, compared to the first one, relies more on the accuracy

of the predictions. It could be seen as working with an old dataset to elaborate new production plan, hence the interest of having a old "big enough dataset". Consequently, in the following we will work with the second regression function considering that the dataset is "big enough".

8.2 The regression function

Here we work with the regression function described just above. We will here develop its construction and its links to its dependencies.

Here is the explanation of the algorithm that summarize the creation of the observations that will be used for the estimation , we will denote by $\boldsymbol{N_o}$, the numbers of required output observations, and by **Pred_Mod** our elaborated prediction model which was trained and validated on $\boldsymbol{D_{\text{a\_pred}}}$, such that $\boldsymbol{D_a} = \boldsymbol{D_{\text{a\_pred}}} \cup \boldsymbol{D_{\text{a\_heuris}}}$ and $\boldsymbol{D_{\text{a\_pred}}} \cap \boldsymbol{D_{\text{a\_heuris}}} = \emptyset$. A simple way to deal with the previous partition of the data, and their selection and restriction, we can vertically concatenate our data. That means our data are ranged with chronological order, the first line correspond to the first hour of a year and the last one correspond to the last hour of another year, which happened after the first one. Let us denote by $|\boldsymbol{D_a}|$, the total number of lines(hours) contained in our dataset, by **prop_pred** the proportion of the dataset used for the training and validation of the prediction model, thus we have : $|\boldsymbol{D_{\text{a\_pred}}}| = \textbf{prop\_pred}|\boldsymbol{D_a}| \in \mathbb{N}$ and $|\boldsymbol{D_{\text{a\_heuris}}}| = (\mathbf{1} - \textbf{prop\_pred})|\boldsymbol{D_a}| \in \mathbb{N}$. Now we have by choosing to split our dataset such that the first part is dedicated to the prediction model and the second part is dedicated to the heuristic : $\boldsymbol{D_{\text{a\_pred}}} = \boldsymbol{D_a}[\mathbf{1} : |\boldsymbol{D_{\text{a\_pred}}}|]$ and $\boldsymbol{D_{\text{a\_heuris}}} = \boldsymbol{D_a}[|\boldsymbol{D_{\text{a\_pred}}}| + \mathbf{1} : |\boldsymbol{D_a}|]$. This is just an easy way split the data, and therefore not necessarily the best one, it will just make the algorithm clearer and more readable. Finally, we denote by **Heuris**, the heuristic that will use the subroutine presented in the subsection 6.3 through the algorithm1.

Algorithm 5 Generation of observations

Require: $D_{\text{a\_heuris}}$, P_r, N_T, θ, N_{res}, N_o, stop_event
Ensure: $Z = (Y, X) \in (\mathbb{N}^{N_T} \times \mathbb{R}^{3\times N_{\text{res}}})^{N_o}$

r_list $\leftarrow$ randint($N_{\text{res}}, |D_{\text{a\_heuris}}| - N_T, N_o$) ▷ a list of N_o random integers $\in [\![\text{N\_res}; |D_{\text{a\_heuris}}| - N_T]\!]$ without replacement
$Z \leftarrow [\,]$
for i in 1:N_o **do**
    Pred_solar_ $\leftarrow$ Pred_Mod($D_{\text{a\_heuris}}$[r_list[i] $- N\_res$, r_list[i]]$_{|\text{solar}}$)
    Pred_wind $\leftarrow$ Pred_Mod($D_{\text{a\_heuris}}$[r_list[i] $- N\_res$, r_list[i]]$_{|\text{wind}}$)
    Pred_park = solar_capacity $\times$ Pred_solar + wind_capacity $\times$ Pred_wind
    Pred_prices $\leftarrow$ Pred_Mod($D_{\text{a\_heuris}}$[r_list[i] $- N\_res$, r_list[i]]$_{|\text{prices}}$)
    Real_solar $\leftarrow$ $D_{\text{a\_heuris}}$[r_list[i] $+ 1, \text{r\_list}[i] + N_T]_{|\text{solar}}$
    Real_wind $\leftarrow$ $D_{\text{a\_heuris}}$[r_list[i] $+ 1, \text{r\_list}[i] + N_T]_{|\text{wind}}$
    Real_park = solar_capacity $\times$ Real_solar + wind_capacity $\times$ Real_wind
    Real_prices $\leftarrow$ $D_{\text{a\_heuris}}$[r_list[i] $+ 1, \text{r\_list}[i] + N_T]_{|\text{prices}}$
    $Y \leftarrow$ Heuris(Pr, θ, Pred_park, Pred_prices, Real_park, Real_prices, stop_event)
    $Z[i] \leftarrow (Y,$ Pred_solar, Pred_wind, Pred_prices)
end for

Now that we have generated our N_o observations, we can get into the regression area, that means using statistical inferences to get the best regression function in terms of transposed costs.

8.3 Estimation of the regression function

Depending on the size of the dataset and on the computation power, we can assume that we have $N_o < 3N_{\text{res}}$, this means that we are in a case of statistics in high dimensions, in other words we have more predictors than observations. So different methods of regression could be tested, for example a decision tree, which allows to select the most significant dimension, or a ridge or lasso regressions, than can also deal with high dimension.

9 Predictions of solar and wind profiles

9.1 Motivation

In our project modeling choice, it is assumed that, at the time of solving the cost minimization problem, a probabilistic space $(\Omega, \mathcal{A}, P)$ is known, along with two prediction random variables $X_{sol}, X_{win} : \Omega \to F_{sol}, F_{win} \subset \mathbb{R}_+^{168}$, representing possible profiles of solar and wind energy production for the upcoming week. Naturally, even with meteorological forecasts, achieving a perfect or negligibly error-prone prediction for these profiles is completely unrealistic. However, considering an in-commensurable variety of improbable profiles leads to no exploitable results. Therefore, our objective in this section is to characterize this probabilistic space as discriminatively as possible concerning wind and solar profiles with non-negligible probability of realization, and to highlight an explicit probability measure on $F_{sol}, F_{win} : P(E) = P(X \in E), E \subset F$.

9.2 Presentation of Different Methods

We only have, in our effort to discriminate between feasible weekly profiles, a large dataset tracing wind and solar energy productions from a German park over the last 5 years. To initiate the study of these time series, we refer to a doctoral work on the *Prediction of global solar radiation time series and photovoltaic energy production*. This work clearly presents the different prediction methods that have yielded promising results.

Naive Models These models simply aim to provide a reference to evaluate the performance of the models used.

Conditional Probability Models These models are often based on the notion of Markov chains.

Seasonal Autoregressive Moving Average Models This popular family assumes that a periodic signal affected by noise can be exhibited in the dataset.

Neural Networks The most used network schemes are multi layer perceptrons, supplemented by networks specialized in time series such as LSTMs.

These prediction methods, except for conditional probability ones, only provide a prediction using an algorithm trained to minimize the error on a training dataset: $\text{Min}_p \ \ err(f(p, x_i), y_i)$ or alternatively

$$\begin{aligned} &\text{Min} \quad \|\epsilon\| \\ &\text{s.t.} \quad f(p, x_i) + \epsilon_i = y_i \quad i \in I_{train} \end{aligned}$$

where p are hyperparameters of the prediction algorithm, y_i is the observed profile at week i, and $f(p, x_i)$ is the prediction made from previous data x_i before week i.

To exploit the results and return to a set of probable profiles, we assume that the noise ϵ obtained after minimization is drawn from a random variable following a centered normal distribution $\mathcal{N}(0, \sigma I)$

with the variance σ to be estimated (we will verify the relevance of this assumption). Thus, we obtain $X_{sol} = f_{sol}(p_{sol}^{opt}, x) + \epsilon_{sol}$ or $X_{win} = f_{win}(p_{win}^{opt}, x) + \epsilon_{win}$ in the case where we would use these single prediction models.

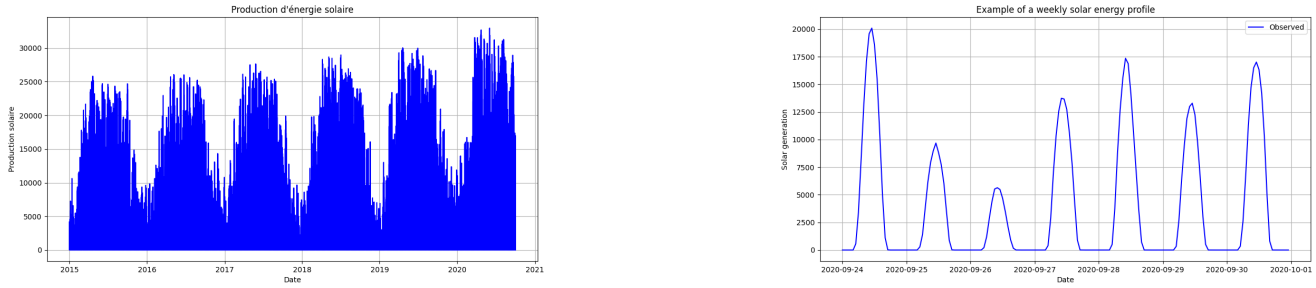
Note that in the constraints of the major cost minimization optimization problem, solar and wind profiles appear only as their sum. However, a quick Python script shows a negative correlation of -0.1749495944282806 between the two profiles. It might therefore be advantageous to focus on predictions of the total energy production profile to take advantage of the non-independence of the two energy sources. Nevertheless, in the following, we will systematically treat the two energy sources separately.

9.3 Chosen Modelings

In order to identify two effective prediction methods to define our probability measures P_{sol} on F_{sol}, P_{win} on F_{win}, we first engage in a quick observation of the two datasets.

Solar Energy

The observation of solar profile data reveals two extremely strong seasonalities: a daily seasonality based on the circadian solar cycle and an annual summer/winter seasonality.
This seasonality would justify at first glance the use of SARIMA models. However, the predictions must



(a) Solar profile on 5 years

(b) Solar profile on a week

Figure 11: Seasonality of solar profile

be generated over 168 successive time steps, which is not at all the purpose of such models generally used for predictions of 2 or 3 time steps. We could only use a SARIMA model in the case where our system is controlled with a few hours' horizon. Moreover, tests performed from a SARIMA model with a one-week horizon immediately show that from 5 prediction time steps, the model produces aberrant results.

We start by using a naive model based on 1 circadian sinusoid and a variable mean for peak height to account for the observed 2 periodicities. The prediction function takes as input the profile of the last 6 days before the start of the prediction period denoted by $\mathbf{x} = (x_k)_{k\in\{1,6\times 24\}}$ (the data is taken for each day from midnight to midnight), calculates the average of the peaks (maximum daily values) over these 6 days $m = \frac{1}{6}\sum_{l=0}^{5} \max\{x_k, k \in \{(l-1)\times 24 + 1, l \times 24\}\}$, and then returns the following profile: $\mathbf{y} = (y_i)_{i\in\{1,7\times 24\}}$ with

$$y_i = \begin{cases} m(sin(\frac{h-h_d}{T}\pi))^+ & \text{if } 0 \leq \frac{h-h_d}{T} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

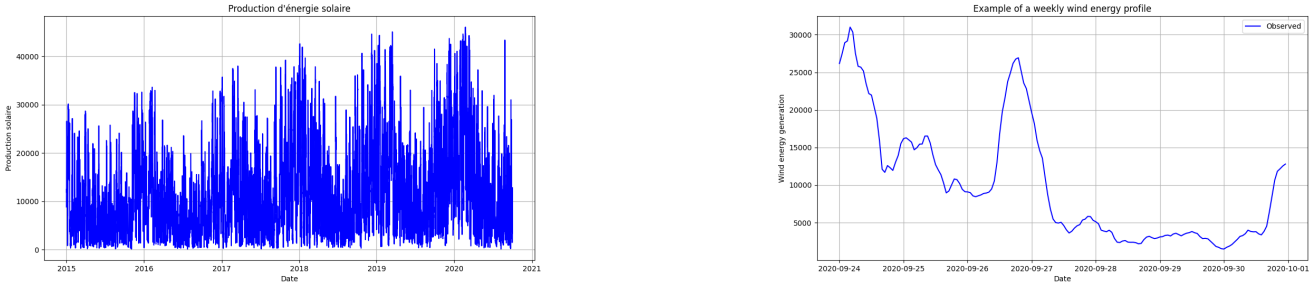
where $h = i \mod 24$, h_d is the sunrise time determined relative to the distance from the beginning of the prediction to January 1st, and T is the day length, determined by the same method.

With a "min max" normalization of the dataset, the obtained quadratic error is 0.0085.
The error obtained by training an LSTM model and a model with 3 fully connected neuron layers with ReLU-type nonlinearities on our dataset are respectively 0.0076and 0.0069 (also taking into account the profile of the last 6 days before the start of the prediction).
The conditional probability models were immediately discarded. Therefore, our solar prediction model is of the form $X_{sol} = f^{NN}(\mathbf{x}) + \epsilon_{sol}$ with $\epsilon_{sol} \sim \mathcal{N}(0, \sqrt{0.0069} \cdot I)$ and $\mathbf{x}$ is the profile of the last 6 days before the start of the prediction. By Shapiro-Wilk test, we verify that the noise $\epsilon_i = \mathbf{y_i} - f^{NN}(\mathbf{x_i})$ obtained on the test dataset has a cumulative distribution function coinciding with the announced distribution.

Wind Energy

Our wind energy profile dataset only shows an annual seasonality, but no daily seasonality. At the weekly level, the values taken by the profile appear to be randomly generated at each time step, except that there seems to be some persistence in the plot.

No single noisy prediction can be relevant, so we focus on a Markov chain model. We define our state



(a) Wind profile on 5 years

(b) Wind profile on a week

Figure 12: Seasonality of wind profile

space E by regular discretization between the minimum and maximum values observed in the dataset. We construct the transition matrix $P(e_n, e_m) = \frac{N_{e_n \to e_m}}{N_{e_n}}$ where the transition probability from state e_n to state e_m is given by the frequency of the transition $e_n \to e_m$ divided by the number of visits to state e_n. The probability of a weekly profile $\mathbf{y} = (y_i)_{i \in \{1,7\times 24\}}$ is then given by $P(\mathbf{y}) = \prod_{i=0}^{7\times 24-1} P(y_i, y_{i+1})$, where y_0 is the state occupied at the previous time step before the prediction period.

The relevance of using a Markov chain with memory, where the state space consists of the values taken at d successive time steps, can be discussed.

9.4 Integration of Predictions and Error Management

The integration of the random nature of possible profiles over the upcoming week is done by the Monte Carlo Method. The hydrogen production cost minimization program is launched with the cost function being the average of the costs given by a number N of solar and wind profiles generated from the previously calculated probabilities: $C(\pi) = \frac{1}{N} \sum_{s=1}^{N} c(\pi, (\mathbf{y_s^{sol}}, \mathbf{y_s^{win}}))$. A solar profile is generated by drawing a Gaussian noise added to the prediction $f^{NN}(\mathbf{x})$, while a wind profile is generated starting from the initial state y_0 and randomly drawing a following state according to the law given by the transition matrix.

It seems interesting to capture the influence of "the error between the predictions made for the upcoming week compared to the profiles actually observed during this week" on "the additional cost generated by the implementation of the optimal policy on real profiles that do not correspond to the different predicted profiles," which is referred to here as the transposition cost, defined by

$$F\big(\text{Real\_park, Real\_price,Opt\_park}(x_s)\text{,Opt\_Ener\_Market}(x_s)\big) = \sum_{t=1}^{T} [\text{Opt\_park}(x_s)_t + \text{Opt\_Ener\_Market}(x_s)_t - \text{Real\_park}_t]^+ \text{Real\_Price}_t \tag{14}$$

If we try, for different prediction error measures $\|\mathbf{y_{real}} - \mathbf{y_{pred}}\|$ to highlight any correlation between prediction error and transposition cost, the results are inconclusive. However, according to the formula itself, it can be noted that the transposition cost is proportional to the negative part $\sum_i (y_i^{pred} - y_i^{real})^+$ when the optimal policy encourages very little purchasing over the week, and the transposition cost is proportional to the cumulative prediction error $\sum_i y_i^{pred} - y_i^{real}$ if the optimal policy encourages purchasing a lot over the week.

9.5 Using the predictions in the dynamic solver

We can integrate the predictions we get using the previously presented model in the dynamic solver (when computing the expected value, we use predicted profiles)

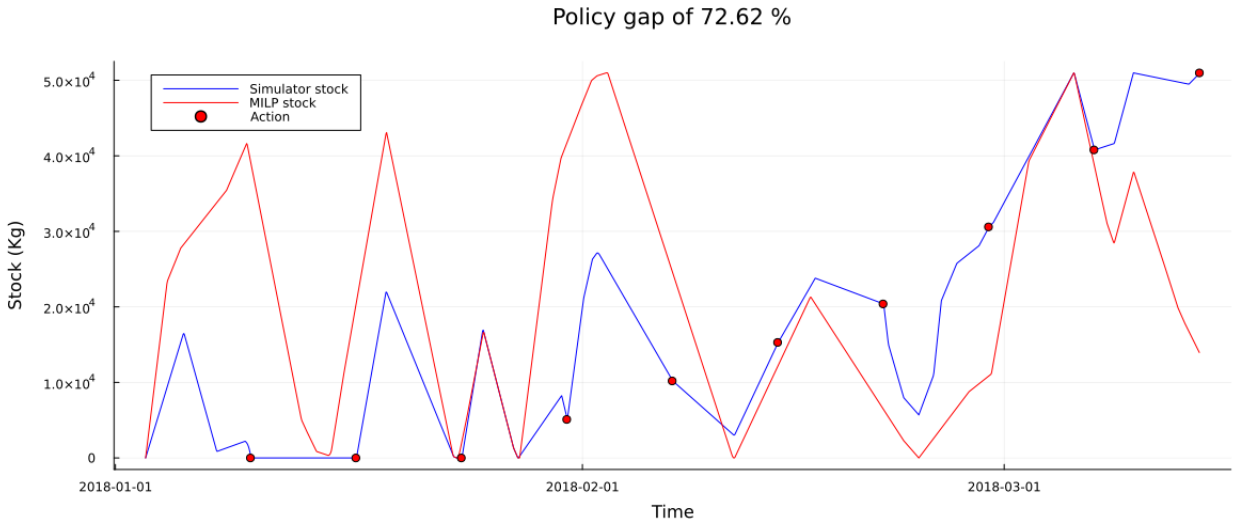


Figure 13: Using the predictions in the dynamic solver, $N_{ev} = 1$

Because of computation time constraints, we were not able to tune the predictors and adapt them. Still, seeing as we use only one prediction profile for each expected value compute, the results we get are not that far from the optimal , and follow the global shape of the optimal piloting (while testing different parameters, we easily got gaps of > 1000 %). With enough time to tweak and adapt the models, we could certainly improve those results and thus the usability of the policy we compute.

Conclusion

As we saw in this report, using dynamic programming to compute policies is a good approach of our initial problem, as it allows us to get a quite good approximation of the optimal cost. More tweaking of the parameters, especially with respect to the integration of the prediction models, could definitely lead us to better results, as well as evidently more compute power.
Concerning the part with the heuristic and the statistical concepts, we saw that having non-zero constrained final levels of hydrogen could help to decrease the transposed cost, but this happen with a specific set of predictions, indeed another set gave opposite conclusion. This fact led to the idea to combine the model of prediction with the heuristic that is supposed to find quickly a good approximation of the best group of constraints for the final levels of hydrogen. The desire of this combination resulted in the conceptualisation of the the regression function, which was not computed, partly due to too heavy computational costs. Different choices of modelling were made in this part, which may not be realistic or at least applied in reality. Nevertheless, the modelling of the regression function was designed in order to be quite flexible to changes on the modelling, such as new parameters for the infrastructure or new constraints and new costs for the linear optimization problem. The idea was to give a coherent and flexible way/protocol to study the impact of this particular constraints on the real cost, which may eventually be different from the transposed cost depending on the choices in the modelling that were made.

Bibliography

[1] 2022 electricity atb technologies and data overview. https://atb.nrel.gov/electricity/2022/index. Accessed: 2024-01.

[2] Open power system data - hourly time series. https://doi.org/10.25832/time_series/2020-10-06. Accessed : 2024-01.

[3] Nicolas Bessin, Edgar Duc, and Baptiste Vert. Code repository for the project. https://github.com/air-liquide-IMI-project/H2Modelling.

[4] Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023.