Reservoir optimization and Machine Learning methods

Xavier WARIN [*]

June 27, 2022

Abstract

After showing the efficiency of feedforward networks to estimate control in high dimension in the global optimization of some storages problems, we develop a modification of an algorithm based on some dynamic programming principle. We show that classical feedforward networks are not effective to estimate Bellman values for reservoir problems and we propose some neural networks giving far better results. At last, we develop a new algorithm mixing LP resolution and conditional cuts calculated by neural networks to solve some stochastic linear problems.

1 Introduction

In the industry, storages are reservoirs used to manage a stock of fuel to satisfy a given objective function. Hydraulic reservoirs use water to produce electricity and the goal of the manager is to supply energy to satisfy a demand at the lowest cost. Gas storage is a special case of storages with a valuation achieved by buying and selling gas on the market. Another example of storages are batteries where electricity is directly injected or withdrawn leading to systems that can be valuated directly facing the market or that can be used to secure a global electrical system.
When the number of reservoirs in low, traditional Dynamic Programming methods are generally used. A typical example is the gas storage which is optimized and hedged using this kind of approach [War12]: the storage level is discretized on a grid, and continuation values are calculated by regression either using the Bellman values at the following time step [TVR01] or using the cash generated at the following dates according to the classical Longstaff Schwartz approach [LS01]. Theses regressions are coupled with linear interpolation in the Bellman values or the cash generated at the different points of the grid. The method can be used for non linear problems only in low dimension for two reasons. The most obvious one is the computing time that explodes with the dimension and computer clusters are necessary to face this computing cost even with a number of reservoir limited to 3 or 4. The second reason which is in fact the first limiting one is the need to store in memory the Bellman values needed by the software. This has led to the development of algorithms splitting the Bellman values on the memory of the different nodes of the computer cluster (see [MVW07] and the StOpt library [Gev+18] for a recent implementation). But even with this kind of approach it is impossible to optimize problems with more than 4 or 5 reservoirs. Another pitfall that managers encounter even in dimension one, as it is the case for gas storage optimization facing the gas market, is the loss of concavity observed due to the regressions and interpolations while calculating conditional expectations even when the solution is known to be concave with respect to the storage level.
In most countries having many dams, dynamic programming methods are not directly used and generally the SDDP (Stochastic Dual Dynamic Programming) method [PP91] is used to manage the dams using a cut approximation of the Bellman values that are concave for a linear objective function with respect to the stocks level in reservoirs. Transition problems are solved using LP solvers with Bellman cuts as a upper estimation of the terminal value. When uncertainties have to be included in the state breaking concavity of the Bellman values, trees are generally introduced and cuts have to be generated at each node of the tree as explained in [PP91]. Another approach consists in generating conditional cuts using regressions [AW20]. In all the cases, forward resolutions (exploration of the possible uncertainties and storage levels visited) and backward resolutions (adding cuts at the levels visited in the forward resolution) give an iterative method that converges [Sha11]. One advantage of this method is that, as cuts

[*]EDF R&D, FiME xavier.warin at edf.fr

are used to generate an approximation of the Bellman values, concavity is preserved with respect to the storage level and the marginal cost of the system (i.e. the derivative of the Bellman value with respect to the storage level) is decreasing with the storage level. However the method may be very slow to converge if the number of transition step is high and it only can be used for linear or special quadratic problems.

In this article, we will investigate the use of neural networks to valuate reservoirs possibly in high dimension. This approach is old as it has been first used for gas storage in [BE+06] at a time where no automatic differentiation software were available and maximization of the profit generated by the storage had to be achieved using a gradient method calculating explicitly the gradients. In this work, feedforward networks are used to approximate the control (gas injection- withdrawal) at each time step and the solution obtained is compared to some trees method used to describe uncertainties. This part of the previous article has been at that time largely ignored by the scientific community. Recently, this kind of representation of the control has been used to solve some BSDE problems [HJW18] leading to many works on this approach.
Using some Dynamic Programming methods, [Bac+21] proposed some algorithm using neural networks and give some numerical results for the valuation of a storage:

- The first one "Control Learning by Performance Iteration" has Longstaff Schwartz flavor: the control at the current date is approximated by a neural network and the controls calculated at the previous time steps (so at the next time steps as the resolution is backward) are reused to estimate the expectation of the objective function starting with a randomized initial state at the current date. In practice, this method can only be used for a very low number of time steps as the cost of recalculating the objective function using the previously calculated controls is very high.
- The second method "Hybrid now" first consists in calculating the control at the current time step by a first optimization by neural networks and then estimate the Bellman values at a current time step by regression by a second optimization problem using a second feedforward network.

At last in a very recently article, [Cur+21] studies the valuation and hedging of a gas storage using a single optimization approximating the control at each time step by some neural networks as in [BE+06]. They also propose to "merge" the network between different time steps (so introducing a dependence on time in a network shared between different time steps). This kind of approach merging all time steps (so using a single neural network shared between all time steps) is the one proposed in [FMW19] for risk valuation or [CWNMW19] for BSDE resolution: they show that it gives better stabilized results on these problems.
The present article studies first the "Global Valuation" method (GV) of one or many storage using the global approach used in [BE+06] and [Cur+21] testing different formulations and network to represent the control.
This type of resolution is impossible to use on real problem involving optimization for example on the global year on an hourly basis for one storage, or the global year even on a daily basis when many storages are linked together: this is for example the case for the valuation of a high number of batteries with a management having an influence on the price of electricity as it would the case for some windmill fields with batteries.
In this case, we have to split the problem and use a modification the "Hybrid now" scheme solving the problem backward but estimating the control not a single time step but for a whole period. This scheme will be called the "Global Split Dynamic Programming" method (GSDP) and "Hybrid now" scheme is a special case were the transition problem is solved on a single time step. This scheme is studied on some difficult test cases and the use of nearly optimal network to represent the Bellman values is studied.
The main findings of this article are the following ones:

- When the underlying process of the problem is Markovian, it is possible to optimize reservoirs very accurately with simple feedforward networks with a small number of layers and neurons for the control even in high dimension using the GV method but it is necessary to use a network for each time step and the "merged" network has to be avoided as it leads to very poor result when the case is really stochastic. When the underlying process is not Markovian, we propose a combination of a LSTM network with some feedforward networks to solve the problem and we show the effectiveness of the methodology.

- The use of the GSDP method results in a loss of optimality due to the calculation of the Bellman values by regression with classical feed forward networks. We show that with a single storage, it is possible to get reasonable results using a feedforward network by increasing the number of layers and neurons. Using 5 or 10 storages optimized together, we show that feedforward are not effective to estimate bellman values. In order to use the GSDP method in high dimension, we develop a network inspired by [AXK17] and show that it gives reasonable results in all dimension tested even with a small number of layers and neurons. When the problem is concave with respect to the storage level, we show that the network proposed in [AXK17] can be used to get good results while preserving concavity and we show that this network is surpassed by an extension of the new GroupMax network proposed in [War22].
- As the GroupMax network generate cuts, we develop a new a algorithm GMCSDP to solve stochastic linear problem using a dynamic programming approach based on LP resolutions as the SDDP method does but with only a single backward resolution. This approach permit to avoid the problem of storing the Bellman values one a grid and LP can solved in parallel using threads. We show the effectiveness of the method on a linear case.

In the whole article we will focus on a maximization of the gain of management in expectation. This choice is driven but the fact that, in practice, it is the main concern associated to this kind of management. Introducing some hedging strategies as proposed in [Cur+21] leads to the need to use a risk function to discriminate an optimal strategy. Another important point associated with this choice is the fact that we can implement multi storage optimization problems that can easily be reduced to a one storage optimization permitting to get a reference by the classical dynamic programming approach using regressions. Therefore we can check that neural network really permit to solve some rather high dimension problems.

2 The global approach for storage optimization

On first recall for convenience that a feedforward network with K hidden layers and m neurons is a non linear operator ϕ $\mathbb{R}^{d_0} \longrightarrow \mathbb{R}^{d_1}$ defined by the following recurrence:

$$z_0 = x \in \mathbb{R}^{d_0}, \tag{1}$$
$$z_{i+1} = \rho(\sigma_i z_i + b_i), 0 \leq i < K, \tag{2}$$
$$\phi(x) = \hat{\rho}(\sigma_K z_K + b_K), \tag{3}$$

where :

- $\sigma_1 \in \mathbb{R}^{m \times d_0}$, $\sigma_i \in \mathbb{R}^{m \times m}$ for $i = 1, \dots, K-1$, $\sigma_K \in \mathbb{R}^{d_1 \times m}$,
- $b_i \in \mathbb{R}^m$, for $i = 0, \dots, K-1$, $b_K \in \mathbb{R}^{d_1}$,
- ρ, $\hat{\rho}$ are activation functions (Elu, Relu, tanh etc..) applied component wise.

The set of all matrix and vector coefficients of σ_i, b_i, $i = 0, \dots, K$ define the set θ of the parameters of the network.
In a first part, we test different approximation and formulation for a linear storage problem. We then extend our results in dimension above taking a non linear problem linking the management of the different storages.

2.1 On a linear problem in dimension one

We suppose we want to manage a storage (gas storage, battery) on a commodity market (gas, electricity) where the commodity follows the HJM model

$$\frac{dF(t,T)}{F(t,T)} = e^{-a(T-t)}\sigma dW_t \tag{4}$$

where W_t is a one dimensional Brownian motion defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. The spot price is then $S_t = F(t,t)$.

The characteristics of the storage are the withdrawal C_W and injection C_I capacities (both taken positive) during one time step Δt, its capacity Q_{Max} and the initial stock position Q_{Init}.
We define the objective function for N optimization dates $t_i = i\Delta t$, for $i = 0, \dots, N-1$:

$$J(U) = -\mathbb{E}[\sum_{i=0}^{N-1} S_{t_i} u_i] \tag{5}$$

for $U = (u_i)_{i=0,N-1}$ in the set $\mathcal{U}$ of the non anticipative admissible strategies such that

$$\begin{aligned} 0 \leq Q_j := Q_{init} + \sum_{i=0}^{j-1} u_i \leq Q_{Max}, \quad & \text{for } j = 0, \dots, N, \\ -C_W \leq u_i \leq C_I, \quad & \text{for } j = 0, \dots, N-1. \end{aligned} \tag{6}$$

We want to maximize the expected gain associated the storage management

$$J^* = \sup_{U \in \mathcal{U}} J(U) \tag{7}$$

As the problem is Markovian in (S, Q) where Q is the storage level, we can introduce as in [BE+06] a feed forward networks $\phi_i^{\theta_i}$ with parameters θ_i per time step i as an operator from $\mathbb{R}^2$ to $\mathbb{R}$ approximating a transformation of the control u_i. There are many ways to deal with the constraints imposed on the level of the storage (Q has to stay positive and below Q_{max}): among them clipping the control, penalization of the objective function are possible but the best approach (we won't report results on less effective approaches) consists in using the [Cur+21] approach. First we introduce for a given i in $0, \dots, N-1$:

$$\hat{C}_I^i = ((Q_i + C_I) \wedge Q_{max}) - Q_i, \quad \hat{C}_W^i = Q_i - ((Q_i - C_W) \vee 0). \tag{8}$$

Then $\phi_i^{\theta_i}$ is a function of (S, Q) using an tanh activation function for ρ and a sigmoid activation function with values in $[0, 1]$ for $\hat{\rho}$. At last the control is approximated as:

$$-\hat{C}_W^i + (\hat{C}_W^i + \hat{C}_I^i)\phi_i^{\theta_i}$$

Noting $\theta = (\theta_i)_{i=0,N-1}$, we approximate the optimal storage management by solving

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}[\sum_{i=0}^{N-1} S_{t_i}(-\hat{C}_W^i + (\hat{C}_W^i + \hat{C}_I^i)\phi_i^{\theta_i}(S_{t_i}, Q_i))] \tag{9}$$

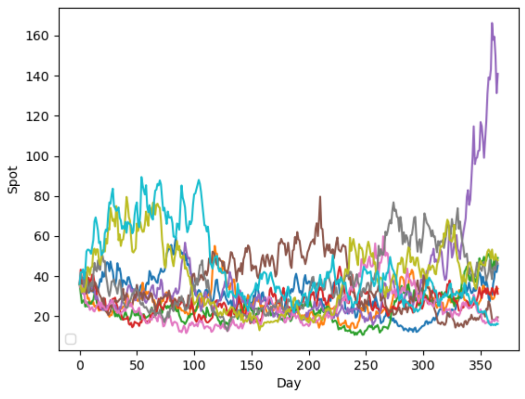
where Q_i follows (6), $\hat{C}_W^i$ and $\hat{C}_I^i$ are given by (8) and the dynamic of F follows (4).

A second version consists classically in introducing a single neural network ϕ ("merged" network) with parameters θ as a function of (t, F, Q) and the optimization (9) is modified as:

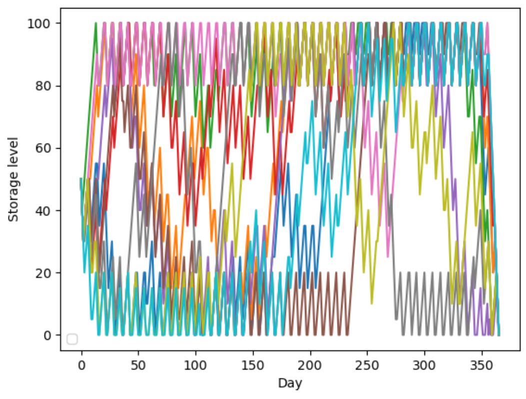
$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}[\sum_{i=0}^{N-1} S_{t_i}\big(-\hat{C}_W^i + (\hat{C}_W^i + \hat{C}_I^i)\phi^\theta(t_i, S_{t_i}, Q_i)\big)] \tag{10}$$

We use a classical stochastic gradient descent ADAM method in Tensorflow [Aba+15] and use some normalized data for F and Q as input of the neural network.
For the test case, we suppose that we optimize on $N = 365$ days a storage with one decision per day ($\Delta t = 1$). The price parameters (expressed in days) $\sigma = 0.08$, $a = 0.01$. The initial forward curve presents seasonal and weekly oscillations as in the energy market and is given by $F(0,T) = 30 + 5\cos(\frac{2\pi T}{N}) + \cos(\frac{2\pi T}{7})$. As for the storage, we take $C_W = 10$, $C_I = 5$, $Q_{Max} = 100$, $Q_{Init} = 50$. A reference is calculated using the StOpt library [Gev+18] by dynamic programming using adaptive linear regression [BW12] and cash flow interpolations as exposed in [War12]. In optimization, 100 basis functions and $1e6$ trajectories are used to optimize the control with regressions. The parameters are taken such that the solution is very stochastic and the problem hard to solve by dynamic programming explaining while we took such parameters for the resolution. As the solution is bang bang [BE+06], we use 20 grid points to

discretize the storage and only bang bang controls are tested leading to a very rapid estimation. Then a simulation using the Bellman values calculated in optimization is achieved. The value in optimization obtained is equal to 4938, while the value in simulation taken as a reference is equal to 4932.



Prices



Optimal storage trajectories

Figure 1: 10 spot and optimal management trajectories.

Using 2 hidden layers with 11 neurons, we train the problem (9) and (10) using mini batch with size 200, 100000 iterations for the gradient descent, and an initial learning rate equal to $2E-3$. Results obtained after training using 200000 trajectories are given in table 1 using 10 runs. Minimal/Maximal is the minimal/maximal value obtained on the 10 runs, while Average is the average value obtained.

	Maximal	Minimal	Average	Min diff with DP
One network per day (9)	4925	4914	4922	7
A singe network (10)	3944	1795	3702	988

Table 1: Neural network valuation with 10 runs.

Results are excellent with a network by day but very bad with a single network. Results with a single network don't change while increasing the number of layers and neurons. Using a network per time step, we can check that results remains in all case very good while decreasing the volatility σ.

2.2 Linear problem increasing the dimension

To get a linear problem with a higher dimension, we suppose that we have M similar storages to manage each one with a strategy $(u_0^j, \dots, u_{N-1}^j)$ for $j = 1, \dots, M$. We note $Q_i = (Q_i^j)_{j=1,\dots,M}$ where Q_i^j is the level in storage j at date t_i, $U = ((u_i^j)_{i=0,N-1})_{j=1,M}$ and the function to maximize is:

$$J^M(U) = -\mathbb{E}[\sum_{j=1}^{M} \sum_{i=0}^{N-1} S_{t_i} u_i^j] \tag{11}$$

and

$$J^{M,*} = \sup_{U \in \mathcal{U}} J^M(U) \tag{12}$$

where all strategies satisfy a flow equation similar to (3). Similarly to the previous section, we introduce a neural network per time step depending on the current prices and the different storage levels. Then a transformation of the network leads to the control. Therefore the network with parameters θ_i per time step i as an operator from $\mathbb{R}^{1+M}$ to $\mathbb{R}^M$ approximating a transformation of the controls $(u_i^1, \dots, u_i^M)$. Using the same activation function as in the previous section, we have to optimize:

$$\theta^* = \operatorname*{argmin}_\theta \mathbb{E}[\sum_{i=0}^{N-1} S_{t_i}(-\hat{C}_W^i + (\hat{C}_W^i + \hat{C}_I^i)\phi_i^{\theta_i}(S_{t_i}, Q_i)).\mathbb{1}_M], \tag{13}$$

where now the $\hat{C}_W^i$ and $\hat{C}_I^i$ are now vectors in $\mathbb{R}^M$ with each component satisfying an equation similar to (8).
In this part, we test two networks:

- First, the classical feedforward network ϕ previously introduced,
- Secondly, as the solution is symmetrical, we test the DeepSet network [Zah+17] (with the same parameters for the number of layers and neutrons as originally proposed by the authors) permitting to impose that the control satisfy the symmetry:

  $$u^l(S, Q^1, \dots, Q^l, \dots, Q^m, \dots, Q^M) = u^m(S, Q^1, \dots, Q^m, \dots, Q^l, \dots, Q^M)$$

  for all $(Q^1, \dots, Q^M)$ state positions in the storages. This network has proved to be more effective than feedforward networks but for very high dimension PDE arising from particular approximation of some mean field problems [Ger+21].

For each storage, we take the same storage characteristics as before and we have $J^{M,*} = MJ^{1,*}$. Training and simulation parameters are the same as in previous section except the number of neurons taken equal to $10 + M$. On figure 2 we plot the results obtained by the two networks. Results obtained by the classical feedforward networks are already optimal and DeepSet networks are not interesting in so small dimensions.

Network	M	Maximal	Minimal	Average	Min diff with DP
feedforward	3	4926	4915	4921	6
feedforward	10	4931	4918	4925	1
DeepSet	3	4896	4882	4889	35
DeepSet	10	4904	4891	4896	28

Table 2: Neural network valuation divided by the dimension in the linear case. 10 runs.

2.3 Results on a non linear case

In order to get reference results for a non linear case, we now suppose that the price is not exogenous anymore and that the impact is proportional to $\frac{P}{M}$ leading to modify the function to maximize:

$$J^M(U) = -\mathbb{E}[\sum_{j=1}^{M} \sum_{i=0}^{N-1} (S_{t_i} + \frac{P}{M} \sum_{l=1}^{M} u_i^l) u_i^j] \tag{14}$$

This kind of modeling of a price impact is necessary for example while managing some batteries when the amount of energy managed represents a rather important part of the energy available on the market. Using the same methodology as before, taking $P = 0.2$ and the same parameters as in the previous sections, we can first for $M = 1$ get a reference with a very thin discretization of the command and the grid storage using dynamic programming with the StOpt library. The value obtained by classical regressions is equal to 3802 in optimization and a value in simulation taken as reference is equal to 3796. The solution of (12) satisfies again $J^{M,*} = MJ^{1,*}$ and we give $\frac{J^{M,*}}{M}$ obtained by the previously defined feed forward network for this non linear case in table 3.

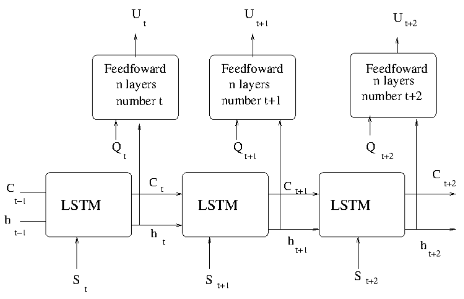
M	Maximal	Minimal	Average	Min diff with DP
1	3794	3780	3788	7
5	3799	3789	3794	2
10	3797	3784	3791	5

Table 3: Neural network valuations with ten run divided by the dimension in the non linear case.

Once again results are very good in dimension 1 to 10 on this very stochastic case as shown on table 3.

2.4 Extension in the non Markovian case

When the price, or in a more general framework the uncertainties are not Markovian, it is possible to extend the previous feedforward network to deal with this feature.
As the optimal control is, at a date t, a function of the whole history of price $(S_u)_{u \le t}$ and the current position in the storage, the idea is to use a recurrent network such a LSTM network [HS97] to deal with the price dependency. At each time step, the output of the LSTM network (with the price as input) and the position in the storages are used as the input of a feedforward network giving the control. The figure 2 represents an unrolled version of the LSTM linked with the feedforward at each time step.



U t
U t+1
U t+2
Feedfoward n layers number t
Feedfoward n layers number t+1
Feedfoward n layers number t+2
Q t
Q t+1
Q t+2
C t-1
h t-1
LSTM
C t
h t
LSTM
C t+1
h t+1
LSTM
C t+2
h t+2
S t
S t+1
S t+2

Figure 2: Unrolled LSTM with feedforward to approximate control.

In order to test this network we now suppose that the future price (4) is replaced by (15).

$$\frac{dF(t,T)}{F(t,T)} = \sum_{i=1}^{3} e^{-a_i(T-t)} \sigma_i dW_t^i, \tag{15}$$

where now $W_t = (W_t^1, W_t^2, W_t^3)$ is a three dimensional Brownian motion.
We still consider the linear problem (5) with the same characteristics as before but with $(\sigma_1, \sigma_2, \sigma_3) = (0.04, 0.028, 0.023)$ and $(a_1, a_2, a_3) = (0.01, 0.005, 0.0033)$. In table 4, we compare the results obtained using the feedforward taking as input the three risk factors of the price model $(\sigma_i \int_0^t e^{-a_i(t-s)} dW_s^i)_{i=1,3}$ and the storage level with the results obtained with the LSTM-Feedforward network where the LSTM network takes the price as input and has 50 units as output. Using dynamic programming with the StOpt library we could get a value of 4300 achieving an optimization regressing with [BW12] method in dimension 3 with $1e7$ trajectories and 10^3 meshes. We could not refine the results due to memory problems.

Network	M	Maximal	Minimal	Average
feedforward	1	4332	4322	4328
feedforward	5	4333	4320	4327
feedforward	10	4334	4318	4329
LSTM-feedforward	1	4285	4277	4280
LSTM-feedforward	5	4284	4273	4279
LSTM-feedforward	10	4285	4268	4279

Table 4: Neural network valuation divided by the dimension in the linear case for the non Markovian case (10 runs).

Results are very good with a very small loss of accuracy compared the feedforward network. We notice that the results obtained with the feedforward networks are slightly above the results obtained by our not converged classical regression method.

3 Global Split Dynamic Programming method

The Global method proposed in the previous section appears to be very effective but may be impossible to implement if the number of dates is too important: memory issues appears and another approach has to be used. This leads to the development of a combination of the Hybrid-Now method of [Bac+21] and the global method of [BE+06]. Suppose that the objective is

$$J(U) = \mathbb{E}[\sum_{i=0}^{N-1} f(t_i, S_{t_i}, U_i)] \tag{16}$$

where the control U_i is a vector of size M, with some flow constraints as in (6) but applied component by component on the control. Suppose that we want to solve (12) and that we split the N dates in $N = \sum_{l=1}^{L} \hat{N}_l$ with $\hat{N}_l \in \mathbb{N}^*$ for $l = 1, \dots, L$. Algorithm 1 permits to solve the problem (12).

```
Algorithm 1: GSDP method
  Output: Estimates the Bellman values at dates $t_0$, $t_{\sum_{l=1}^{\tilde{l}} \hat{N}_l}$, for $\tilde{l} = 1, L-1$, and all optimal
   controls.
  $\tilde{i} = N$
  $VB^L = 0$
  for $l = L, \dots, 1$ do
     $\tilde{i} = \tilde{i} - \hat{N}_l$
     Introduce $\hat{N}_l$ feedforward networks $\phi_k^{\theta_k}$ on $\mathbb{R}^{M+1}$ with values in $\mathbb{R}^M$ with sigmoid activation
      for the output
        $$\theta^* := (\theta_0^*, \dots, \theta_{\hat{N}_l-1}^*) = \operatorname*{argmin}_\theta \mathbb{E}[\sum_{i=0}^{\hat{N}_l-1} f(t_{\tilde{i}+i}, S_{t_{\tilde{i}+i}}, U_i^\theta) + VB^l(S_{\tilde{i}+\hat{N}_l}, Q_{\hat{N}_l})] \tag{17}$$
      such that $Q_0 \sim U[0, Q_{max}]^M$ and for $0 \le i < \hat{N}_l$:
        $$\hat{C}_I^i = ((Q_i + C_I) \wedge Q_{max}) - Q_i, \qquad \hat{C}_W^i = Q_i - ((Q_i - C_W) \vee 0)$$
        $$U_i^\theta = -\hat{C}_W^i + (\hat{C}_W^i + \hat{C}_I^i)\phi_i^{\theta_i}(S_{t_{\tilde{i}+i}}, Q_i), \quad Q_{i+1} = Q_i + U_i^\theta \tag{18}$$
      Introduce a neural network $\psi^\kappa$ with identity output activation function with parameters $\kappa$
        $$\kappa^* = \operatorname*{argmin}_\kappa \mathbb{E}[\Big(\sum_{i=0}^{\hat{N}_l-1} f(t_{\tilde{i}+i}, S_{t_{\tilde{i}+i}}, U_i^{\theta^*}) + VB^l(S_{\tilde{i}+\hat{N}_l}, Q_{\hat{N}_l}) - \psi^\kappa(S_{t_{\tilde{i}}}, Q_0)\Big)^2], \tag{19}$$
      where (18) is satisfied and $Q_0 \sim U[0, Q_{max}]^M$
     $VB^{l-1} = \psi^{\kappa^*}$
```

In this algorithm the use of a feedforward may seem natural to solve (19) as it was proposed for the hydrid-now method in [Bac+21]. We take our linear test case in dimension 1 and test the use of this network for ψ^κ to solve (19) for different number of L values (only $\hat{N}_0$ may differ from the $\hat{N}_l$ for $l > 0$ that are all equal :$\hat{N}_l = \hat{N}_m$ for $l > 0$ and $m > 0$). As for the neural networks used for the control in (17) we keep the same characteristics as in the previous sections.
Using different number of neurons and layers to approximate the Bellman values, in table 5 we give the results obtained with the best of 3 runs taking 100000 gradient iterations with an initial learning rate equal to $5E-3$ still with the ADAM method. The activation function ρ in (3) is a Relu function as it gives better results than tanh activation function and it gives similar results to *Elu* activation function.

L	m	$\tilde{L}$	solution	Min diff with DP
4	11	2	4900	32
13	11	2	4816	116
53	11	2	4389	543
4	30	3	4899	33
13	30	3	4834	98
53	30	3	4633	299

Table 5: GSDP method for the one dimensional linear case using a feed forward with a number of neuron m, and a number of layers $\tilde{L}$ to solve (19). We take the best of 10 runs.

There is a loss of optimality increasing as L increases. For $L = 53$ we have to take at least 3 layers and 30 neurons to avoid too much loss of accuracy. Using more layers or neurons can very slightly improve the results and now we take 5 layers with 20 neurons for this feedforward network and now test the previous linear case and the non linear one from section 2.2 and 2.3 keeping $L = 53$ and letting the dimension increase. We report results on table 6.

Test case	M	max	min	average	min error
Linear	1	4651	4543	4606	280
Linear	5	4352	4252	4179	579
Linear	10	2547	1831	2151	1248
Non linear	1	3687	3646	3664	108
Nonlinear	5	3074	2122	2529	721
Nonlinear	10	2039	1379	1165	1756

Table 6: Feed forward results $\frac{J^{M,*}}{M}$ with the GSDP method for linear and non linear cases with 5 layers, 20 neurons for (19), $L = 53$ and letting the dimension M vary. We use 10 runs and report the best (max), the worst (min) results, and at last the error for the best result obtained.

The degradation of the result obtained with the dimension is obvious and even in dimension 5 the error obtained is far too important. The treatment of the storage levels has to be made different from the uncertainty regression and it leads to the development of more adapted networks. Then the input of the network is split into two parts:

- All uncertainties that are regressed are be treated using a feedforward network,
- A special treatment of the dimension associated to the storage where only interpolation is needed is proposed.

We propose to use three different networks.

3.1 A first network ψ^A preserving concavity

As on this case, the solution is concave with respect to storage we can use a modification of the [AXK17] network avoiding penalization given by the recursion (20). Supposing that the input of the neural network $\tilde{x} = (x, y) \in \mathbb{R}_{d_0}$ where we have concavity in $y \in \mathbb{R}^k$,

$$\begin{aligned} u_{i+1} =& \tilde{\rho}(\tilde{W}_i u_i + \tilde{b}_i) \\ z_{i+1} =& \rho([W_i^{(z)} \otimes (W_i^{(zu)} u_i + b_i^{(z)})]^+ z_i + \\ & W_i^{(y)}(y \circ (W_i^{(yu)} u_i + b_i^{(y)})) + W_i^{(u)} u_i + b_i), \quad \text{for } i \leq K \\ \psi^A =& z_{K+1}, \quad u_0 = x, \quad z_0 = 0 \end{aligned} \tag{20}$$

where ρ is a concave non increasing activation function that we will take equal to minus Relu, $\circ$ denote the Hadamard product, $\otimes$ is applied between a matrix $A \in \mathbb{R}^{m \times n}$ and vector $B \in \mathbb{R}^n$ such that $A \otimes B \in \mathbb{R}^{m \times n}$ and $(A \otimes B)_{i,j} = A_{i,j} B_j$. As stated in [AXK17], concavity of the solution is given by the characteristics of ρ and the fact that the weight before z_i is positive in (20). Using m_x neurons for the non concave part of function and m_y neural networks for the convex part of the network, $\tilde{W}_0 \in \mathbb{R}^{m_x \times d_0 - k}$, $\tilde{W}_i \in \mathbb{R}^{m_x \times m_x}$ for $i > 0$, $W_i^{(zu)} \in \mathbb{R}^{m_y \times m_x}$ for $i > 0$, $W_i^{(z)} \in \mathbb{R}^{m_y \times m_y}$ for $j < K$, $W_K^{(z)} \in \mathbb{R}^{1 \times m_y}$ as the output is a scalar function. We don't detail the size of the different matrix $W^{(y)}$, $W^{(yu)}$, $W^{(u)}$ and the different bias that are obvious.

In all experiments, $\tilde{\rho}$ is the ReLU activation function.

3.2 A second network ψ^{AD} removing the concavity constraints

We naturally modify the previous one removing the constraints on concavity by

$$\begin{aligned} u_{i+1} =& \tilde{\rho}(\tilde{W}_i u_i + \tilde{b}_i) \\ z_{i+1} =& \rho(W_i^{(z)}(z_i \circ (W_i^{(zu)} u_i + b_i^{(z)})) + \\ & W_i^{(y)}(y \circ (W_i^{(yu)} u_i + b_i^{(y)})) + W_i^{(u)} u_i + b_i), \quad \text{for } i \leq K \\ \psi^{AD} =& z_{K+1}, \quad u_0 = x, \quad z_0 = 0 \end{aligned} \tag{21}$$

Removing the concavity constraints, we get a network that can be used even for non concave/convex problems. In all experiments $\tilde{\rho}$ and ρ are ReLU activation functions.

3.3 The GroupMax network ψ^{GM} using cuts when the solution is concave

The GroupMax is a network developed very recently [War22] combining ideas in [ALG19], [TB21] and the ones in [AXK17] but permitting to have cuts to represent a concave solution. When the function is only concave with respect to y, the following network is proposed in [War22] generating cuts conditional to x:

$$\begin{aligned} u_0 =& x, \quad z_0 = 0 \\ u_{i+1} =& \tilde{\rho}(\tilde{W}_i u_i + \tilde{b}_i) \\ z_{i+1} =& \rho([W_i^{(z)} \otimes (W_i^{(zu)} u_i + b_i^{(z)})]^+ z_i + \\ & W_i^{(y)}(y \circ (W_i^{(yu)} u_i + b_i^{(y)})) + W_i^{(u)} u_i + b_i), \quad \text{for } i \leq K-1 \\ \psi^{GM}(x,y) =& \hat{\rho}([W_K^{(z)} \otimes (W_K^{(zu)} u_K + b_K^{(z)})]^+ z_K + \\ & W_K^{(y)}(y \circ (W_K^{(yu)} u_K + b_K^{(y)})) + W_K^{(u)} u_K + b_K) \end{aligned} \tag{22}$$

where all matrices involved have the same size as the matrices in (21) except that the matrix $W_K^{(z)}$ is in $\mathbb{R}^{m_y \times m_y}$.
In (22), the $\tilde{\rho}$ is a classical activation function such as ReLU and in order to get conditional cuts to approximate the solution, we take $\hat{\rho}$ as an activation function working on the whole vector:

$$\hat{\rho}(x) = \min_{i=1,\dots,d} x_i \quad \text{for } x \in \mathbb{R}^d.$$

The ρ is defined grouping the elements of the vector as in the GroupSort network [ALG19]. Supposing that $x \in \mathbb{R}^{m_y}$, $G \leq m_y \in \mathbb{N}^*$ the group size such that $\tilde{m} = \frac{m_y}{G} \in \mathbb{N}^*$ representing the number of groups, ρ maps $\mathbb{R}^{m_y}$ to $\mathbb{R}^{\tilde{m}}$ such that:

$$\rho(x)_i = \min_{j=1,\dots,G} x_{(i-1)G+j}, \quad \text{for } i = 1, \dots, \tilde{m}.$$

In [War22], it is shown that this network generates conditional cuts with respect to x. In all experiments, $\tilde{\rho}$ is a ReLU activation function.

3.4 Numerical results

We test the three networks on the linear and the non linear case. It is obvious that in the linear case, the Bellman value is concave with respect to the storage level. In the non linear case, concavity is still present

[GLP15]. All the Bellman values obtained by the three networks are not very sensitive to the number of layers and neurons. For the three networks used to approximate Bellman values, we take $m_x = 10$ and 3 hidden layers, taking $m_y = 20$ for the two first networks and $m_y = 40$ for the GroupMax network. We keep the same parameters as in the previous sections to estimate the controls.

M	Network	max	min	average	min error
1	ϕ^A	4679	4618	4645	252
1	ϕ^{AD}	4798	4738	4771	133
1	ϕ^{GM}	4810	4777	4795	122
5	ϕ^A	4352	3949	4179	579
5	ϕ^{AD}	4482	4318	4399	449
5	ϕ^{GM}	4641	4519	4601	290
10	ϕ^A	4027	3724	4027	904
10	ϕ^{AD}	4151	3890	4031	780
10	ϕ^{GM}	4425	4316	4351	506

Table 7: Result $\frac{J^{M,*}}{M}$ of the Linear case with the GSDP method with L=53 for the different networks. 10 runs.

M	Network	max	min	average	min error
1	ϕ^A	3585	3540	3558	210
1	ϕ^{AD}	3687	3646	3664	108
1	ϕ^{GM}	3688	3663	3676	107
5	ϕ^A	3444	3206	3412	351
5	ϕ^{AD}	3538	3278	3434	257
5	ϕ^{GM}	3633	3589	3614	162
10	ϕ^A	3456	3044	3288	339
10	ϕ^{AD}	3389	3308	3205	406
10	ϕ^{GM}	3556	3482	3556	239

Table 8: Result $\frac{J^{M,*}}{M}$ on the Non Linear case with the GSDP method with L=53 for the different networks. 10 runs.

As the dimension M increases, the variance of the results obtained increases. The GroupMax is clearly superior to other networks. Results are better, the loss of accuracy decreases less quickly with the dimension and the variance of the results obtained is much lower than with the other networks.
We conclude that is optimal to use a number L as low as possible. When the problem is not concave with respect to the storage levels, the ϕ^{AD} network is the best one and when the problem is concave, the cut methodology given by the GroupMax network ϕ^{GM} is the best choice.

4 GroupMax Cut Split Dynamic Programming (GMCSDP) method

The management of reservoirs in high dimension is often achieved with a linear stochastic model. Using a hazard decision framework, the resolution uses a stochastic model where uncertainties are revealed for a period Δt and some decisions are taken on sub intervals of Δt. Using the fact that the Bellman values are concave with respect to the storage levels, at each time step $t_i = i\Delta t$, decisions are taken starting at a level in the reservoirs $Q = (Q_1, \dots, Q_M)$ in the storage problem by solving some LP problem with terminal conditions given by some cuts of the Bellman values at date t_{i+1}. The Bellman values being generally not concave with respect to the uncertainties, cuts are conditional to the uncertainty level. Theses cuts are often given at the node of a scenario tree [PP91] and can also be calculated by regressions as in [AW20].

- When the dimension is low, starting point in the reservoirs for the LP problems are derived by exploring a grid, and a single backward resolution is achieved. This is a dynamic programming method.

- When the dimension becomes higher, a procedure iterating backward resolution and some forward exploration simulations is used. The forward step permits to reveal the reservoir levels of interest to explore during the following backward step that adds new cuts. This iteration procedure is the SDDP of [PP91].

The two methods are developed in some libraries such StOpt [Gev+18] using regressions and trees. These methods are very popular as they permit to treat difficult constraints dealt by the LP solver used to solve the transition problems.

Remark 4.1. *The non linear transition problem with a cost function quadratic concave with respect to the control and the reservoir level can also solved by a quadratic solver using the same methodology but at a higher cost.*

As stated in the introduction, the convergence can be very slow and the stopping criterion be can be tricky to implement especially when the Bellman values are not concave with respect to the uncertainties. The use of dynamic programming methods estimating the cuts on a grid suffers both from the computing time and the memory needed. We propose to use the GroupMax network to estimate the Bellman values by cuts.
We suppose that we want to solve equation the following equation

$$\begin{aligned}
J^* = & \sup_{U=(U_0,U_{N-1})\in\mathcal{U}} \sum_{i=0}^{N-1} \mathbb{E}[f(t_i, U_i, S_{t_i})] \\
\underline{U} \leq & U_i \leq \bar{U}, \text{ with } U_i \in \mathbb{R}^p, \quad i = 0, N-1 \\
X_0 = & \tilde{X} \in \mathbb{R}^q \\
X_{i+1} = & X_i + A_i(S_i)U_i + B_i(S_i) \in \mathbb{R}^q, \quad i = 0, N-1 \\
\underline{X} \leq & X_i \leq \bar{X}, \quad i = 0, N
\end{aligned} \tag{23}$$

where f is linear or quadratic concave with respect to U, S_t is a discrete time Markov process in $\mathbb{R}^d$, A_i a function from $\mathbb{R}^d$ to $\mathbb{R}^{q\times p}$, B_i function from $\mathbb{R}^d$ in $\mathbb{R}^q$ for $i = 0, N-1$.
Based on the dynamic programming principal and using the GroupMax network, we propose to use the

algorithm 2 to optimize (23) where the Bellman values are estimated by cuts.

Algorithm 2: GMCSDP method

```
**Output:**
$VB^{N-1} = 0$
**for** $i = N-1, 1$ **do**
    Introduced a GroupMax neural network $\psi^{GM,\theta}(S,X)$ with parameter $\theta$
        $$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}[(\psi^{GM,\theta}(S_{t_{i-1}}, X) - \mathcal{Q}(S_{t_i}, X))^2]$$
     where
        $$\begin{aligned} \mathcal{Q}(S_{t_i}, X) = \max_{U \in \mathbb{R}^p} & f(t_i, S_{t_i}, U) + \xi \\ \xi \le & VB^i(S_{t_i}, \tilde{X}) \\ \tilde{X} = & X + A_i(S_{t_i})U + B_i(S_{t_i}) \\ \underline{X} \le & \tilde{X} \le \bar{X}, \\ \underline{U} \le & U \le \bar{U}, \end{aligned}$$
     with $X \sim U([\underline{X}, \bar{X}])$, $S_{t_{i-1}}$ sampled from $S_0$ and $S_{t_i}$ sampled from $S_{t_{i-1}}$
    $VB^{i-1}(s,x) = \psi^{GM,\theta^*}(s,x)$
Optimize first time step:
        $$\begin{aligned} \max_{U \in \mathbb{R}^p} & f(0, S_0, U) + \xi \\ \xi \le & VB^0(S_0, \tilde{X}) \\ \tilde{X} = & X_0 + A_0(S_0)U + B_0(S_0) \\ \underline{X} \le & \tilde{X} \le \bar{X}, \\ \underline{U} \le & U \le \bar{U}, \end{aligned}$$
```

To test the algorithm, we suppose that the problem is linear, given by (12), (11) with the flow equation (6). The characteristics of the storages are unchanged. The initial forward curve is given by $F(0,T) = 30 + 4\cos(\frac{2\pi T}{7})$. The price model is still given by (4) but with the parameters $\sigma = 0.3$, $a = 0.16$. We take $N = 42$ and the optimization by the dynamic programming method using the property that the optimal control are bang bang gives a value of 3426 while the value obtained in forward using the optimal control is 3424. Using algorithm 2, we use a GroupMax network with 2 layers (one hidden layer) and a group size equal to 2. The resolution with the ADAM stochastic gradient descent uses a batch size of 200, a number of gradient iterations equal to 15000 with an initial learning rate equal to $5E-3$ and decreasing to $1e-4$. Local optimizations are achieved with the Coin LP solver. Results for $\frac{J^*}{M}$ are given in table 9.

M	m_y	max	min	average	min error
1	8	3362	3317	3335	61
1	10	3370	3321	3349	53
1	12	3355	3334	3345	69
3	8	3207	3113	3163	218
3	10	3225	3173	3202	200
3	12	3233	3166	3194	192
5	8	3063	2758	2913	360
5	10	3012	2729	2885	411
5	12	3052	2799	2980	371

Table 9: GMCSDP results on 10 runs for $\frac{J^*}{M}$. Reference is SDP with regressions.

The accuracy decreases with the dimension while the variance of the result obtained increases. We don't see any clear differences using different m_y values except perhaps in dimension 3, where an increase of m_y seems to give slightly better results.

Remark 4.2. *As the time induced by the LP resolution is highly related to the number of cuts used, we limit the number of layers to 2 ($K = 1$ which permit to have $m_y 2^{\frac{m_y}{2}}$ cuts using m_y neurons). We keep m_x equal to* 8*. As shown in [War22], it is necessary to increase the number of layers to get high accuracy but this leads to very time consuming problems to solve.*

In real industrial problems, constraints between storage reduces the volatility of the system and convergence should be easier to achieve. This approach permits to avoid the memory cost due to the storage of the Bellman functions and only the computing time remains a constraint: parallelization by threads and MPI should permit to reduce efficiently this computing cost.

References

[Aba+15] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[ALG19] Cem Anil, James Lucas, and Roger Grosse. "Sorting out Lipschitz function approximation". In: *International Conference on Machine Learning.* PMLR. 2019, pp. 291–301.

[AW20] Wim van Ackooij and Xavier Warin. "On conditional cuts for stochastic dual dynamic programming". In: *EURO Journal on Computational Optimization* 8.2 (2020), pp. 173–199.

[AXK17] Brandon Amos, Lei Xu, and J Zico Kolter. "Input convex neural networks". In: *International Conference on Machine Learning.* PMLR. 2017, pp. 146–155.

[Bac+21] Achref Bachouch, Côme Huré, Nicolas Langrené, and Huyen Pham. "Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications". In: *Methodology and Computing in Applied Probability* (2021), pp. 1–36.

[BE+06] Christophe Barrera-Esteve et al. "Numerical methods for the pricing of swing options: a stochastic control approach". In: *Methodology and computing in applied probability* 8.4 (2006), pp. 517–540.

[BW12] Bruno Bouchard and Xavier Warin. "Monte-Carlo valuation of American options: facts and new algorithms to improve existing methods". In: *Numerical methods in finance.* Springer, 2012, pp. 215–255.

[Cur+21] Nicolas Curin et al. "A deep learning model for gas storage optimization". In: *arXiv preprint arXiv:2102.01980* (2021).

[CWNMW19] Quentin Chan-Wai-Nam, Joseph Mikael, and Xavier Warin. "Machine learning for semi linear PDEs". In: *Journal of Scientific Computing* 79.3 (2019), pp. 1667–1712.

[FMW19] Simon Fécamp, Joseph Mikael, and Xavier Warin. "Risk management with machine-learning-based algorithms". In: *arXiv preprint arXiv:1902.05287* (2019).

[Ger+21] Maximilien Germain, Mathieu Laurière, Huyên Pham, and Xavier Warin. *DeepSets and their derivative networks for solving symmetric PDEs.* 2021. arXiv: `2103.00838 [math.OC]`.

[Gev+18] Hugo Gevret et al. "STochastic OPTimization library in C++". PhD thesis. EDF Lab, 2018.

[GLP15] Pierre Girardeau, Vincent Leclere, and Andrew B Philpott. "On the convergence of decomposition methods for multistage stochastic convex programs". In: *Mathematics of Operations Research* 40.1 (2015), pp. 130–145.

[HJW18] Jiequn Han, Arnulf Jentzen, and E Weinan. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[LS01] Francis A Longstaff and Eduardo S Schwartz. “Valuing American options by simulation: a simple least-squares approach”. In: *The review of financial studies* 14.1 (2001), pp. 113–147.

[MVW07] Constantinos Makassikis, Stéphane Vialle, and Xavier Warin. “Distribution of a stochastic control algorithm applied to gas storage valuation”. In: *2007 IEEE International Symposium on Signal Processing and Information Technology*. IEEE. 2007, pp. 485–490.

[PP91] Mario VF Pereira and Leontina MVG Pinto. “Multi-stage stochastic optimization applied to energy planning”. In: *Mathematical programming* 52.1 (1991), pp. 359–375.

[Sha11] Alexander Shapiro. “Analysis of stochastic dual dynamic programming method”. In: *European Journal of Operational Research* 209.1 (2011), pp. 63–72.

[TB21] Ugo Tanielian and Gerard Biau. “Approximating Lipschitz continuous functions with GroupSort neural networks”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 442–450.

[TVR01] John N Tsitsiklis and Benjamin Van Roy. “Regression methods for pricing complex American-style options”. In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 694–703.

[War12] Xavier Warin. “Gas storage hedging”. In: *Numerical methods in finance*. Springer, 2012, pp. 421–445.

[War22] Xavier Warin. “The GroupMax neural network approximation of convex functions”. In: *arXiv preprint arXiv:2206.06622* (2022).

[Zah+17] M. Zaheer et al. “Deep Sets”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3391–3401.