Subway stations optimal energy management

P. Carpentier, J.-P. Chancelier, M. De Lara, V. Leclère and T. Rigaut

October 10, 2017

Contents

Abstract

In this practical work, you will have to manage a battery in order to recover subways braking electrical energy and to supply it to a subway station. Your goal is to minimize the cost of energy consumed on the national grid by exploiting optimally this regenerative braking energy. First you will compute the gains realized when the battery is controlled according to an intuitive strategy. Then you will compare two methods, Stochastic Dynamic Programming and Model Predictive Control. Both methods provide a control for the battey at each time step these methods don't use the same amount of information about the past randomness realization. And they don't use their information the same way. You will compare throughout this work different ways to use past information to compute controls aware of the future uncertainties.

1 Problem statement and data

Subway stations represent one third of the Paris subway system energy consumption. There are unexploited energy ressources in subway stations that could be harvested to lower this energy consumption. We could for example recover braking energy from the trains. Currently trains can dissipate their kinetic energy by braking mechanically or convert this energy into electricity. However supply/demand equilibrium have to be ensured on the subway line. Hence a train can brake electrically only if there is a train accelerating/consuming energy nearby. There is therefore a significant amount of potential energy that is currently wasted. We could overcome this problem with an electrical storage.

We consider a subway station equiped with a battery that can charge braking energy and national grid energy on one side (subway line side) and discharge energy to power the subway station which is also powered by the national grid (subway station side). A battery manager has to control this battery in real time. The station demand and the cost of electricity are assumed deterministic. Braking energy available at each time step is uncertain.

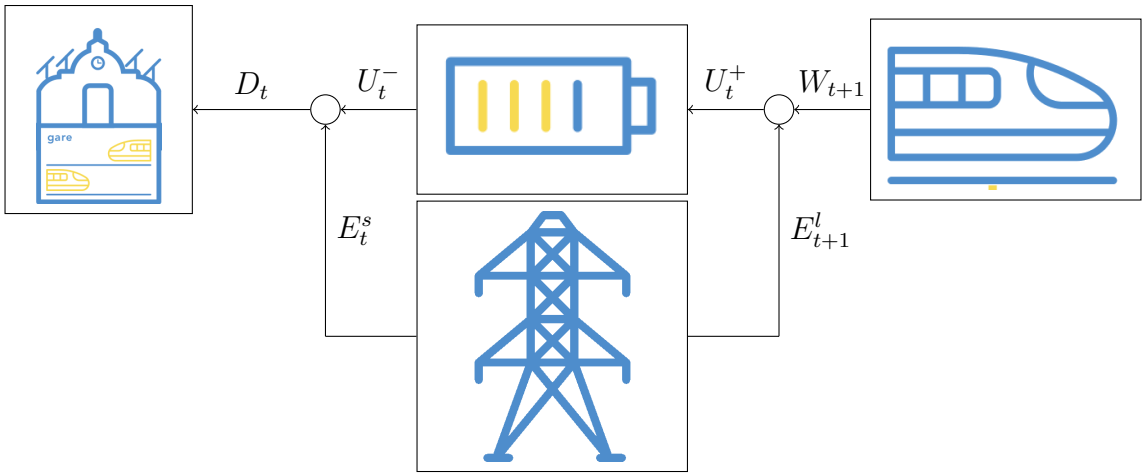

Figure 1: Electrical problem representation

We present hereby the stochastic optimization problem we want to solve:

$$\min_{(U_t(.))_{t\in\{0,..,T\}}} \mathbb{E}[\sum_{i=0}^{T-1} c_t \times (E_t^l + D_t + U_t^-)] \tag{1a}$$

s.t

$$S_{t+1} = S_t + \rho U_t^+ - \frac{1}{\rho} U_t^- \tag{1b}$$
$$D_t - U_t^- \geq 0 \tag{1c}$$
$$U_t^+ \leq E_t^l + W_t \tag{1d}$$
$$E_{t+1}^l \geq 0 \tag{1e}$$
$$S_{min} \leq S_t \leq S_{max} \tag{1f}$$
$$S_0 = x_0 \tag{1g}$$
$$U_{min} \leq U_t \leq U_{max} \tag{1h}$$
$$U_t = \pi_t(W_0, ..., W_{t-1}) \tag{1i}$$

We detail all notations and equations in the following.

1.1 Battery dynamics

The battery can store power coming from the trains or the national grid and supply power to the subway station. It cannot do both things at the same time. We control its state of charge (soc). We model the dynamics of the battery at each time step t by a function f_t we call the dynamic at time t giving equation (1b) :

$$\underbrace{S_{t+1}}_{\text{future soc}} = \underbrace{S_t}_{\text{soc}} + \underbrace{\rho U_t^+}_{\text{effective charge}} - \underbrace{\frac{1}{\rho} U_t^-}_{\text{effective discharge}}$$
$$= f_t(S_t, U_t, W_t)$$

where

- $t \in \mathbb{T} := \{0, 1, \ldots, T-1\}$ is discrete (minutes in this case) and t denotes the beginning of the period $[t, t+1[$. We call T the horizon of the problem.
- S_t state of charge (kWh) of the battery at the beginning of period $[t, t+1[$. It belongs to a set $\mathbb{X} = [S_{min}, S_{max}]$ representing the minimum and maximum state of charge of the battery.
- U_t energy (kWh) of battery charge or discharge during $[t, t+1[$, decided at the beginning of time period $[t, t+1[$. It belongs to a space $\mathbb{U} = [U_{min}, U_{max}]$ modeling ramp limits for the charge or discharge.
- $U_t^- = -\min(0, U_t)$ energy we discharge from the battery
- $U_t^+ = \max(0, U(t))$ energy we charge into the battery
- ρ is the charge and discharge efficiency (%)

The initial state of charge of the battery is called x_0 and appears in the equality constraint (1g)

1.1.1 Variables definition

We define hereby the remaining variables of the problem presented in figure1

- E_t^s energy (kWh) we pay to the national grid to power the station during $[t, t+1[$
- E_t^l energy (kWh) we pay to the national grid to charge the battery during $[t, t+1[$
- D_t energy (kWh) consumed by the subway station during $[t, t+1[$
- W_t braking energy (kWh) available on the subway line during $[t, t+1[$. This quantity is a random variable

We observe that charge/discharge decision U_t is taken at the beginning of $[t, t+1[$, *before knowing* the available braking energy W_t. We are in a *decision-hazard* setting.

1.1.2 Constraints

We have to ensure the supply/demand equilibrium at each node and each time step on the subway station node which leads to equation:

$$D_t - U_t^- = E_t^s \tag{2}$$

On the subway line node we have to ensure supply/demand balance however we are not forced to recover all the regenerative braking energy W_t. We have then an inequality constraint (1d):

$$U_t^+ \leq E_t^l + W_t$$

We prevent the battery manager to sell power to the grid then we have to ensure:

$$\begin{aligned} &E_t^l \geq 0 \\ &E_t^s \geq 0 \quad \text{or} \quad D_t - U_t^- \geq 0 \end{aligned}$$

1.2 Criterion: intertemporal payoff

At each time t we pay to the national grid operator the quantity $E_t^l + E_t^s$ of energy. We assume that the price of electricity c_t per kWh is known in advance as it is negociated and constant during $[t, t+1[$.
At each time t we pay the instantaneous cost:

$$L_t(S_t, U_t, W_t) = c_t(E_t^l + E_t^s)$$

Using equation (2) we can eliminate the variable E_t^s leading to the instantaneous cost:

$$L_t(S_t, U_t, W_t) = c_t(E_t^l + D_t - U_t^-)$$

At the end of the horizon we have no preference concerning the final state of charge of the battery. We introduce then a final cost function $K : \mathbb{X} \to \mathbb{R}$ such as:

$$\forall S_T \in \mathbb{X},\ K(S_T) = 0$$

From 0 to T the payoffs sum up to:

$$\sum_{t=0}^{T-1} c_t(E_t^l + D_t - U_t^-) + K(S_T) \tag{3}$$

The manager wants to minize these payoffs every day without knowing the future availability of braking energy. He will therefore minimize them *on average*, hopefully obtaining the best mean payoff he can obtain after several days. The manager wants to minimize the expected payoff:

$$\mathbb{E}\Big[\sum_{t=0}^{T-1} c_t(E_t^l + D_t - U_t^-) + K(S_T)\Big] \tag{4}$$

The manager wants to solve problem (1). We detail the last constraint (1i) in the following.

1.3 Numerical data and model

1.3.1 Constants

We define, in file *constants.jl*, some numerical data required to model the problem.

```
dt = 15*60. #time step, seconds
T0 = 24*3600 #horizon, seconds
T = floor(Int,T0/dt) #number of stages
hour_stages = floor(Int, T/24.) #number of stages in 1 hour

U_p_max = 100 # ramp upper limit for battery charging, kW
U_m_max = 100 #ramp upper limit for battery discharging, kW
Cap = 40 #battery maximal capacity, kWh
X_max = 0.9 * Cap #battery operational upper capacity, kWh
X_min = 0.3 * Cap #battery operational lower capacity, kWh
rho = 0.95 #efficiency

D = 100+5*rand(T-1)-5*rand(T-1) #demand, kWh;
D = floor(D)
C = 0.06*ones(T-1)*dt/3600. #cost of electricity, euros/kWh;
C[7*hour_stages:9*hour_stages] = 0.08*dt/3600.
C[16*hour_stages:19*hour_stages] = 0.08*dt/3600.
C[9*hour_stages:16*hour_stages] = 0.07*dt/3600.

X_0 = 0.35 * Cap #initial battery state of charge, kWh

U_bounds = [(-U_m_max,U_p_max)] #battery control bounds, kWh
X_bounds = [(X_min, X_max)] #battery state of charge bounds, kWh
```

Question 1 *Comment each line by defining the constant after the* #. *What is the number of time steps of the problem? The number of time steps in one hour? Include the file* constants.jl *in your script or notebook. Plot the cost of electricity as well as the consumption of the station during the day. Compute the cost of the station's energy consumption over the whole time horizon when it consumes exclusively on the national grid.*

1.3.2 Braking energy scenarios

We provide a code function, defined in *scenarios_generation.jl* , that allows to compute a given number of braking energy scenarios over the time horizon.

```
"""
Compute n scenarios of braking energy
```

```
# Arguments
* `Int`:
    the number of scenarios wanted

# Return
* Array{Float64} : array of dim 3 and size (T-1, n, 1)
containing n scenarios of W_{t} realizations

* Array{Float64} : array of dim 3 and size (T-1, n, 1)
containing n scenarios of xi_{t} realizations

"""
function generate_braking_scenarios(n)
```

Question 2 *Include* scenarios_generation.jl *in your script or notebook. Use this function to generate 50 regenerative braking scenarios and save the 2 outputs. The first output contains the braking energy scenarios, we leave the second output until the end. Based on this 50 scenarios what is the average energy available over a day? We will call these 50 scenarios the assessment scenarios thoughout the work. We will refer to the second output as the ξ-assessment scenarios.*

1.3.3 Problem functions declaration

We define here 4 code functions required to model the problem.

```
function cost(t,x,u,w)
    return C[t]*(D[t]+max(0,u[1]-w[1])+min(u[1],0))
end

function constraints(t,x,u,w)
    return (D[t]+min(u[1],0)>=0)
end

function dynamics(t, x, u ,w)
    return [x[1] + dt./3600*(rho*max(u[1],0) + (1/rho)*min(u[1],0))]
end

function final_cost(x)
    return(0)
end
```

Question 3 *What is the number of state variables? What is the number of control variables? Verify that the code functions correspond the real functions of the problem.*

2 Evaluation with given policies

The manager wants to be able to make a decision that takes into account the future uncertainties but that use the information about the past uncertainties realization. This is the meaning of the last constraint (1i) of problem (1) we did not yet explained.

2.1 Non-anticipativity constraint

The constraint $U_t = \pi_t(W_0, ..., W_{t-1})$ states that the decision at time t, U_t, must depend only on the past uncertainties of the problem $W_0, ..., W_{t-1}$.

When the W_t are time independent we can restrict the search to functions of the current state of the problem. The non anticipativity constraint becomes $U_t = \pi_t(S_t)$. When the W_t are not time independent we can still restrict the search to functions of the state. The controls obtained are not guaranteed to be optimal but they may be good enough and are often easier to compute.

An *admissible state feedback policy* $\pi : \mathbb{T} \times \mathbb{X} \to \mathbb{U}$ assigns a charge or discharge instruction $U_t = \pi_t(S_t) \in \mathbb{U}$ to any time $t \in \mathbb{T}$ and to any state of charge $S_t \in \mathbb{X}$, while respecting the problem constraints. Given an admissible policy π and given a braking energy scenario $W(\cdot) = \big(W_0, \dots, W_{T-1}\big)$, we are able to build a battery state of charge trajectory $S(\cdot) := \big(S_0, \dots, S_{T-1}, S_T\big)$ and a charge/discharge control trajectory $U(\cdot) := \big(U_0, \dots, U_{T-1}\big)$ produced by the "closed-loop" dynamics initialized at the initial time 0 by

$$S_0 = x_0$$

and propagated from $t = 0$ up to $t = T - 1$ according to

$$\begin{aligned} U_t &= \pi_t\big(S_t\big) \\ S_{t+1} &= f_t\big(S_t, U_t, W_{t+1}\big) \ . \end{aligned} \tag{5}$$

We also obtain the payoff associated to the braking energy generation scenario $W_1, ..., W_T$:

$$V_0^\pi\big(S_0, W(\cdot)\big) := \sum_{t=0}^{T-1} L_t\big(S_t, U_t, W_{t+1}\big) + K\big(S_T\big) \ , \tag{6}$$

where $S(\cdot)$ and $U(\cdot)$ are given by (5).

The *expected payoff* associated with the policy π is

$$\mathbb{E}V_0^\pi\big(S_0, W(\cdot)\big) \ , \tag{7}$$

where the expectation $\mathbb{E}$ is taken with respect to the product probability $\mathbb{P}$. The true expected value (7) is difficult to compute,[1] and we evaluate it by the *Monte Carlo* method

[1] Note however that this computation is achievable insofar all quantities it involves belong to finite sets.

using N_s braking energy scenarios $\left(W^1(\cdot), \ldots, W^{N_s}(\cdot)\right)$:

$$\frac{1}{N_s} \sum_{s=1}^{N_s} V_0^\pi \left(S_0, W^s(\cdot)\right) . \tag{8}$$

By the law of large numbers, the mean payoff (8) is a "good" approximation of the expected payoff(7) if the number of scenarios is "large enough".

2.2 Simulation of a given heuristic

We propose here a code function providing a heuristic strategy base only on state and time information. At time t this function return a control u_t knowing t and x_t. This code function is then a state feedback. Its idea is to charge during peak hours and discharge during off-peak hours while ensuring the state and control constraints.

```
function my_heuristic(t, x_t)

    U = 0

    #morning peak hours
    if (t>=7*hour_stages)&&(t<=9*hour_stages)

        U = max(rho*(X_min-X_max)/(2*hour_stages+1),
                        rho*(X_min-x_t[1]))*3600/dt

    #evening peak hours
    elseif (t>=16*hour_stages)&&(t<=19*hour_stages)

        U = max(rho*(X_min-X_max)/(3*hour_stages+1),
                         rho*(X_min-x_t[1]))*3600/dt

    else #off-peak hours
        U = min((X_max-x_t)/rho*3600/dt, U_p_max)
    end
    return(U)
end
```

Question 4 *What are the peak and off-peak hours? What is the sign of the control returned by this heuristic according to the time at which it is computed? Explain the logic of this heuristic.*

We provide a code function, in *policy_assessment.jl*, to assess a policy along multiple scenarios knowing the instantaneous cost, the final cost, the dynamic, the scenarios, the initial state of the system and the policy.

```
"""
Assess a policy by simulation of multiple scenarios to compute the expectation of the cost

# Arguments
* `Array{Float64}`:
    initial state of the system
* `Int`:
    number of stage in the problem
* `Function`:
    Instantaneous cost function
* `Function`:
    Final cost function
* `Function`:
    Dynamics function
* `Function`:
    Policy, function of the time and state returning a control
* `Array{Float64}`:
    Noises scenarios along which we want to simulate

# Return
* `Float64`:
    Mean cost obtained by simulation of the heuristic over multiple scenarios
* `Float64`: dim 3 with size (T, dim(X_t), N_scenarios)
    Array containing the state trajectories

"""
function assess_policy(initial_state, stages_number, inst_cost, fin_cost, dyn,
```

Question 5 *Use this function with the previous heuristic on each on the assessment scenarios. What is the average cost of energy consumption on the national grid obtained? Compare with the origanl cost when there was no battery. Plot the battery state of charge for some scenarios. Does it correspond to your previous understanding of the heuristic?*

3 Stochastic Dynamic Programming resolution

Stochastic Dynamic Programming is a method to solve Stochastic Optimal Control problems. It is based on the concept of value functions and the so called Bellman's principle of

optimality. The value function of the problem at time t is defined as:

$$Vt(S_t) := \min_{U_t,\dots,U_T} \mathbb{E}\big[\sum_{k=t}^{T-1} L_k(S_k, U_k, W_{k+1}) + K(S_T)\big]$$
$$\text{s.t} \quad (5)$$

With K the final cost of the problem which is zero in our case. The principle of optimality states that, when the problem noises are independent, we can write the following recursion between value functions:

$$V_t(S_t) = \min_{U_t} \mathbb{E}\big[L_t(S_t, U_t, W_{t+1}) + V_{t+1}\big(f_t(S_t, U_t, W_{t+1})\big)\big] \tag{9}$$

In practice we use equation (9) in two phases: an *offline* stage where we use it to compute value functions, and an *online* stage where these functions are used to compute a control.

Offline phase: computation of value functions
For t from T to 0 compute

$$V_t(S_t) = \min_{U_t \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, U_t, W_{t+1}) + V_{t+1}\big(f_t(S_t, U_t, W_{t+1})\big)\big] \tag{10}$$

Online: computation of a control
At time t, knowing the state of the system S_t compute U_t:

$$U_t \in \arg\min_{U \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, U, W_{t+1}) + V_{t+1}\big(f_t(S_t, U, W_{t+1})\big)\big] \tag{11}$$

In practice we compute the value functions on a discretized state space $\mathbb{X}^d$. As it is not possible to iterate over a continuous set. Similarly we solve problems (19) and (20) by discretizing $\mathbb{U}$ and testing every control in $\mathbb{U}^d$ the discrete space obtained.

3.1 Markovian assumption, state and control discretization

In order to compute the expectation in problems (19) and (20) we assume that the noises are time independent and use 1000 braking energy scenarios to compute the marginal laws of the noises at each time step. We do not use the assessment scenarios in order not to bias the results. The discrete marginal laws are obtained by quantization as Monte Carlo should be avoided as much as possible with Dynamic Programming.

```
#we build the marginal laws
w_laws = build_noises_laws(N_W, 10);
```

We choose the size of the discretization step for the state variable and the control variable.

```
U_steps = [2.] # controls discretization, kW
X_steps = [1.] # states discretization, kWh
```

Question 6 *What is the number of operations required to compute the value functions of the problem?*

3.2 Value functions offline computation

We provide a function to compute the value functions of the problem.

```
V_sdp = StochDynamicProgramming.solve_DP(battery_management_sdp_model,
                                                                    params_sdp, 1);
```

Question 7 *Plot the value function at time 3 over the state of charge of the battery using the function provided in* plotting_functions.jl. *Why is it decreasing? Plot the value function at time 4300. Why does it display a plateau?*

3.3 Using value functions for online control computation

We provide a function, in file *sdp_policies.jl* to generate a control at time t knowing the previous uncertainty realization, the state of the system and the time step.

```
"""
Obtain a control for the battery management problem at time t
knowing state x_t using value functions

# Arguments
* `Int`:
    time step t
* `Array{Float64}`:
    state of the system at time t

# Return
* Array{Float64}:
control computed by the policy

"""
function get_control_sdp_x(t, x_t)
```

This function is a policy that can be plugged in our simulator.

Question 8 *Simulate battery control using value functions along the assessment scenarios. What is the average cost obtained? Compare with the previous methods. Plot the battery state of charge during the day. Do you notice an important discrepancy with the heuristic simulation?*

4 Model Predictive Control

Model Predictive Control is a popular method to solve Stochastic Optimal Control problems arising in many engineering fields (electrical, mechanical...). Its popularity comes from its

simplicity and accessibility. It consists in making a forecast to get rid of all the future uncertainties at each time step. We can then solve a deterministic problem which provides a control trajectory. We apply the first control. At the next time step we compute a new forecast and we reoptimize.

This algorithm has no offline phase.

Online: computation of a control
At time t, knowing the state of the system S_t and the past uncertainties $W_0, .., W_{t-1}$ compute a forecast $\bar{w}_t, .., \bar{w}_{T-1}$ and compute U_t:

$$U_t \in \underset{U_t \in \mathbb{U}}{\arg\min} \min_{U_{t+1},..,U_{T-1}} \Big[\sum_{k=t}^{T-1} L_k\big(S_k, U_k, \bar{w}_{k+1}\big) + K\big(S_T\big) \Big] \tag{12}$$

4.1 Deterministic problem resolution

We provide a function to solve a deterministic problem knowing a scenario, the initial state and the initial time step. This function is in the file *deterministic.jl*.

```
"""
Compute the solution of the problem where noises are replaced by nominal values

# Arguments
* `Array{Float64}`:
    initial state of the system
* `Int`:
    initial time step at which we solve the problem
* `Array{Float64}`:
    Noises scenario to replace uncertainties

# Return
* `JuMPmodel`:
JuMP model modelling the deterministic problem
* Float64:
objective value obtained
* Array{Float64}:
trajectory of the battery state of charge obtained
* Array{Float64}:
trajectory of the control obteined

"""
function solve_anticipative_problem(X_0_loc, t0, stages_horizon, forecast)
```

Question 9 *Use this function to solve the deterministic problem over the whole horizon with initial state x_0 and initial time step 0 for each of the assessment scenarios. Why is the average cost lower than the heuristic and dynamic programming one?*

4.2 Model Predictive Control simulation

We provide a function to compute a forecast at time t for the next time steps up to the rolling horizon length knowing only the state at time t.

This function is used in the MPC policy as it consists in making a forecast of the future time steps and solve a deterministic problem.

```
function classic_mpc_policy(t, x_t)
    forca = compute_forecast(t, T)
    _, _, _, u = solve_anticipative_problem(x_t[1], t, T-t+1, forca)
    return [u[1,1,1]-u[1,2,1]]
end
```

Question 10 *Compute a simulation with the mpc policy for each of the assessment scenarios. Compare with the previous results.*

5 Taking into account non-Markovian structure

We could improve the Model Predictive Control results. Indeed MPC performances depend significantly on the forecast quality. Here we use only information about the state of the system at time t to compute a forecast assuming all the noises are time independent. We the use the previous marginal laws to compute the forecast.

The scenario generator we provided follows the following rules:

- there is no braking energy during the night as subways are not circulating then $\forall t \in \{T_{stop}, ..., T_{start}\},\ W_t = 0$
- during operating hours we model regenerative braking randomness by an autoregressive process of order 1:

$$\forall t \in \{0, .., T_{start}\} \cup \{T_{stop}, .., T\},\ W_t = aW_{t-1} + \xi_t \tag{13}$$

  with $(\xi_t)_{t\in\mathbb{T}}$ a sequence of gaussian white noises.

5.1 Improving MPC forecast

The forecast computed at each time step in the MPC strategy could be based on this model and on the last uncertainty realization. The MPC policy is not anymore a state feedback it lies in the class of mappings:

$$\pi_t : \mathbb{X} \times \mathbb{W} \to \mathbb{U} \tag{14}$$

We provide a MPC policy using this information

```
function ar_mpc_policy(t, x_t, w_t)
    forca = compute_ar_forecast(t, w_t, T, 100)
    _, _, _, u = solve_anticipative_problem(x_t[1], t, T-t+1, forca)
    return [u[1,1,1]-u[1,2,1]]
end
```

Question 11 *Compute a simulation with this mpc policy for each of the assessment scenarios. Compare with the previous results.*

We can build such kinds of policies in a dynamic programming framework as well.

5.2 Using past informations online in classical SDP

We consider the value functions computed at section ??. During the online phase we can compute the expectation with respect to W_t by exploiting the stochastic process structure. We know here that W follows an autoregressive process (or zero during night hours). We know then that: $\mathbb{E}[g(W_t)|W_{t-1} = w_{t-1}] = \mathbb{E}[g(aw_{t-1} + \xi_t)]$. Knowing the law of ξ_t we can therefore estimate the conditional expectation at time t.

Offline phase: computation of value functions
Ignore the dependence between the noises. We keep the same marginal laws for the random variables. For t from T to 0 compute

$$V_t(S_t) = \min_{U_t \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, U_t, W_{t+1}) + V_{t+1}\big(f_t(S_t, U_t, W_{t+1})\big)\big] \tag{15}$$

Online: computation of a control
At each time step t, observing state S_t and previous uncertainty realization w_t compute U_t:

$$U_t \in \arg\min_{U \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, U, aw_{t-1} + \xi_t) + V_{t+1}\big(f_t(S_t, U, aw_{t-1} + \xi_t)\big)\big] \tag{16}$$

We provide a policy of this form using the value functions we generated earlier.

```
"""
Obtain a control the battery management problem at time t
knowing state x_t and noise w_{t-1} using value functions

# Arguments
* `Int`:
    time step t
```

```
* `Array{Float64}`:
    state of the system at time t
* `Array{Float64}`:
    realization of the noises at time t-1

# Return
* Array{Float64}:
control computed by the policy

"""
function get_control_sdp(t, x_t, w)
```

Question 12 *Compute a simulation with this policy for each of the assessment scenarios. Compare with the previous results.*

5.3 State augmentation

Using noises correlation information with value functions computed with time independence assumption mitgh be inefficient in practice.

When the randomness is an order 1 autoregressive process it is possible to improve the stochastic dynamic programming results by augmenting the state with 1 state variable. W follows an AR(1) process. We can then take W_t as a state variable, $\xi(\cdot)$ being the noises of the problem. We just add the following dynamic in the problem (1):

$$W_{t+1} = aW_t + \xi_t \tag{17}$$

$W(\cdot)$ being now a state variable and $\xi(\cdot)$ the exogenous noises process. We call the new dynamic F_t such that:

$$F_t(S_t, W_t, U_t, \xi_t) = (S_t + \rho U_t^+ - \frac{1}{\rho} U_t^-, aW_t + \xi_t) \tag{18}$$

Offline phase: computation of value functions
For t from T to 0 compute

$$V_t(S_t, W_t) = \min_{U_t \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, W_t, U_t, \xi_t) + V_{t+1}\big(F_t(S_t, W_t, U_t, \xi_t)\big)\big] \tag{19}$$

Online: computation of a control
At time t, knowing the state of the system S_t compute U_t:

$$U_t \in \arg\min_{U \in \mathbb{U}} \mathbb{E}\big[L_t(S_t, W_t, U_t, \xi_t) + V_{t+1}\big(F_t(S_t, W_t, U_t, \xi_t)\big)\big] \tag{20}$$

We define a new instantaneous cost code function, a new dynamic, a new state discretization, a new control discretization and new noises laws.

```
function cost_2_vars(t,x,u,w)

    return C[t]*(D[t]+max(0,u[1]-x[2])+min(u[1],0))

end

function dynamics_2_vars(t, x, u ,w)

    return [x[1] + dt./3600*(rho*max(u[1],0) + (1/rho)*min(u[1],0)),
         ((t>=5*hour_stages)||(t<=hour_stages))*max(0,0.8*x[2]+w[1])]

end

X_steps_2_vars = [4., 50] # states discretization, kWh

X_bounds_2_vars = [(X_min, X_max), (0, 400)]

w_laws_2_vars = build_xi_noises_laws(10)
```

Question 13 *What is the number of state variables? What is the number of control variables? What is the difference between the previous instantaneous cost and this one? Same question with the dynamics.*

5.4 Value functions generation and forward simulation

We build another stochastic dynamic programming model. We can generate the value functions of the problem with 2 state variables. Now the noises are truly independent.

```
V_sdp_2_vars =
StochDynamicProgramming.solve_DP(battery_2_vars,params_sdp_2_vars, 1);
```

We provide a function to compute a control online using these value functions. This is a state feedback with the augmented state. It is therefore a function of S_t and W_t.

Question 14 *Using the ξ assessment scenarios that were generated at the beginning, corresponding to our previous assessment scenarios. Assess the policy. Display the average cost obtained.*

5.5 Benchmark

Question 15 *Compare all the methods by filling the following table. Discuss the results.*

Policy	**Offline time**	**Online time**	**Costs w/ ref scenario**
Heuristic			
SDP			
MPC			
MPC w/ w_t			
SDP w/ w_t			
SDP w/ 2 states			

Table 1: Benchmark