Optimal control of a domestic microgrid

SOWG

October 10, 2017

Contents

Abstract

1 Problem statement, formalisation and data

1.1 Problem statement

We consider here a house equipped with a thermal hot water tank, a combined heat and power generator (CHP) and a battery. This house has two systems to satisfy its energetical needs: a thermal system for heating, and an electrical system to satisfy its electrical demand. These two systems are coupled together via the CHP, rendering the optimal control difficult to find.

The CHP is a generator which produces both electricity and heat while burning gas. The power of the CHP can not be modulated: it produces always the same amount of heat and electricity.

A battery can store the potential surplus of electricity (especially when demand is low), and the house is connected to the grid to import electricy if necessary. The heat produced by the CHP is stored in a hot water tank. Hot water is withdrawn from this tank to satisfy thermal demand.

We aim to decrease the operational costs while taking into account that:

- the electrical price depends upon time, as shown in Figure 2,
- electrical and thermal demands are highly variable (the evolution of these demands are displayed in Figure 3).

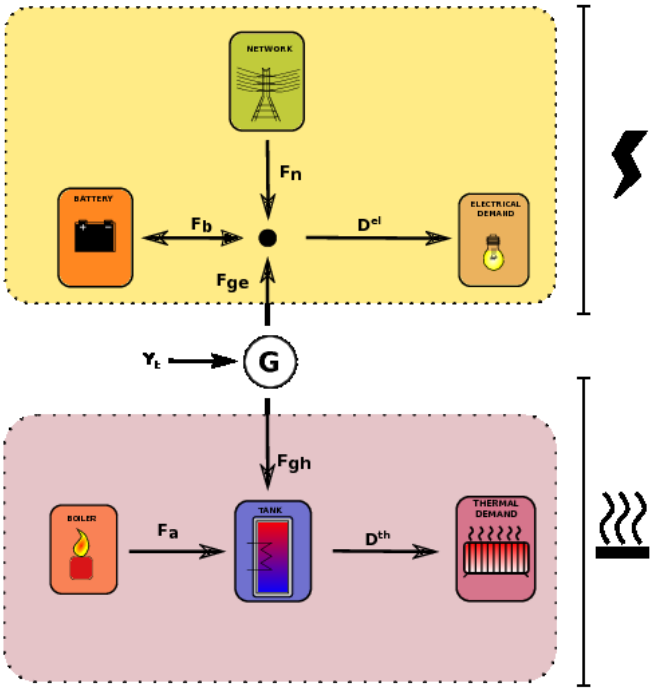

Figure 1: Different systems constituting the domestic microgrid.

1.2 System's dynamic

We model our system with discrete linear equations, stating the evolution of the system between two timesteps t and $t+1$. Controls are computed at the begining of the time interval $[t, t+1[$, and are applied all along this interval. That gives a discrete time optimal control problem.

We control here the system during one day, at a 15mn timestep.

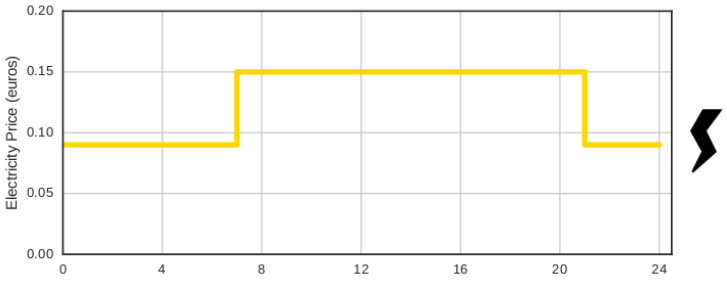

Figure 2: Electrical price

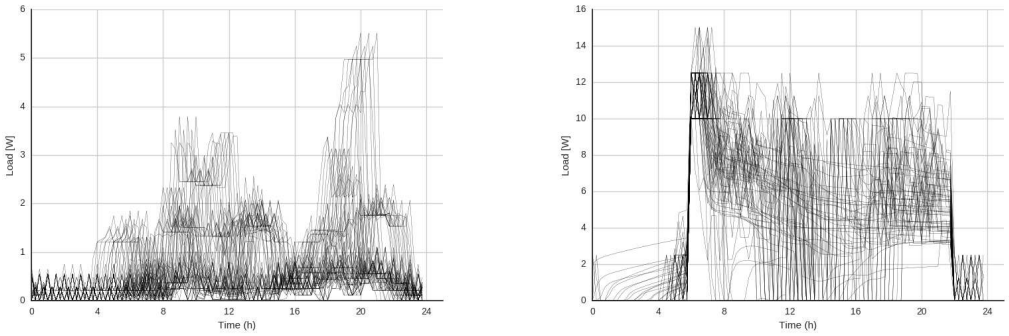

Figure 3: Different scenarios for demand

1.2.1 Battery dynamic

B_t is the energy stored inside the battery at time t. The battery's dynamic is the following:

$$B_{t+1} = B_t + \Delta t \; F_{B,t} \; . \tag{1}$$

Battery's constraints The battery's level must remain greater than a given level otherwise its life expectancy decreases much more quickly. Usually, it is specified that this level remains between 30% and 90% of the maximum level of charge. The charge and discharge are also bounded by a maximum rampage rate ΔB. These two constraints write:

$$B^\flat \leq B_t \leq B^\sharp \; . \tag{2}$$

and

$$-\Delta B \leq B_{t+1} - B_t \leq \Delta B \tag{3}$$

1.2.2 Hot water tank dynamic

We have two inputs for the hot water tank:

- the heat $F_{gh,t} = (1-\alpha)P_{chp,t}$ produced by the CHP,

- the heat $F_{A,t}$ produced by the auxiliary burner,

and one output: the thermal demand of the house D_t^{th}.

We denote the tank's level H_t, denoting the energy stored inside the tank at time t (in kWh). This level is bounded:

$$H^\flat \leq H_t \leq H^\sharp \,. \tag{4}$$

The hot water tank is modelled with:

$$H_t^+ = H_t + \Delta t\big[(1-\alpha)P_{chp,t} + F_{A,t} - D_t^{th}\big] \,, \tag{5a}$$

$$F_{th,t+1} = \min\big[\max\left(0,\ H^\flat - H_t^+\right) - H^\flat + H^\sharp,\ 0\big] \,, \tag{5b}$$

$$H_{t+1} = H_t + \Delta t\big[(1-\alpha)P_{chp,t} + F_{A,t} - D_t^{th} + F_{th,t+1}\big] \,. \tag{5c}$$

All these equations are designed so as the constraints 4 are satisfied for all demand D_t^{th}.

The recourse variable $F_{th,t+1}$ is penalized afterward in the cost function.

1.3 Criterion: operational costs

Electrical exchange We denote $F_{N,t}$ the electricity imported from the grid at time t.

The balance equation between production and demand in the upper electric part in Figure 1.1 writes:

$$\alpha P_{chp,t} + F_{N,t} = D_t^{el} + F_{B,t} \tag{6}$$

The electricity imported from the network is being paid at a tariff π_t^{el}, which depends upon time. The price scenario $\left(\pi_{t_i}^{el}, \ldots, \pi_{T_f}^{el}\right)$ is supposed to be known in advance.

Operational costs Costs between $[t, t+1[$ are:

$$L_t(P_{chp,t}, F_{A,t}, F_{N,t}, F_{th,t+1}) = \underbrace{\pi^{gas}P_{chp,t}}_{\text{CHP}} + \underbrace{\pi^{gas}F_{A,t}}_{\text{aux burner}} + \underbrace{\pi^{th}|F_{th,t+1}|}_{\text{uncomfort}} + \underbrace{\pi_t^{el}F_{N,t}}_{\text{network}} \,. \tag{7}$$

In (7), costs of different devices are considered:

- the cost to use the CHP, that is the gas we burn to produce heat and electricity: $\pi^{gas}P_{chp,t}$,
- the cost to use the auxiliary burner $\pi^{gas}F_{A,t}$,
- the cost to import electricity from the grid $\pi_t^{el}F_{N,t}$,
- the virtual cost of uncomfort if thermal demand is unsatisfied: $\pi^{th}F_{th,t+1}$.

1.4 Optimization criterion

We can now write the optimization criterion:

$$\min_{u} \mathbb{E}\Big[\sum_{t=0}^{T-1} L_t(P_{chp,t}, F_{A,t}, F_{N,t}, F_{th,t+1})\Big] \tag{8}$$

s.t.

$$\begin{aligned} B_{t+1} &= B_t + \Delta t F_{B,t} \\ B^\flat \leq B_t &\leq B^\sharp \\ \Delta B^\flat \leq B_{t+1} - B_t &\leq \Delta B^\sharp \\ H_{t+1} = H_t + \Delta t\big[(1-\alpha)P_{chp,t} + F_{A,t} - D_t^{th} + F_{th,t+1}\big] \\ H^\flat \leq H_t &\leq H^\sharp \end{aligned}$$

We recall the different notations we used previously in Table 1.

State variables		
B_t	Battery level	kWh
H_t	Tank level	kWh
Control variables		
$P_{chp,t}$	CHP power	kW
$F_{B,t}$	Energy exchanged with the battery	kW
$F_{A,t}$	Production of auxiliary burner	kW
Noises		
D_t^{el}	Electrical demand	kW
D_t^{th}	Thermal demand	kW
Parameters		
π^{el}	Electricity price	euros
π^{gas}	Gas price	euros
π^{th}	Price of uncomfort	euros

Table 1: Notations

2 Simulation and evaluation of given policies

The states, controls and perturbations are now denoted:

$$x_t = (B_t, H_t) \tag{9a}$$

$$u_t = (Y_t, F_{B_t}, F_{A,t}) \tag{9b}$$

$$w_t = (D_t^{el}, D_t^{th}) \tag{9c}$$

We suppose that states belong to a static set $x_t \in \mathbb{X}$, so do control $u_t \in \mathbb{U}$.

A demand scenarios is a sequence of demands from time t_0 up to $T_f - 1$.

$$w(\cdot) = \left(w_0, \cdots, w_{T_f-1}\right), \tag{10}$$

where for all t, $w_t = (D_t^{el}, D_t^{th})$.

$w(\cdot)$ is supposed to be the realization of a stochastic process $(W_t)_{t=0}^{T_f}$.

2.1 Evaluation of a policy

An *admissible policy* $\gamma : \mathbb{T} \times \mathbb{X} \to \mathbb{U}$ assigns a control $u = \gamma(t, x) \in \mathbb{U}$ to any time $t \in \mathbb{T}$ and to any state $x \in \mathbb{X}$,

Given an admissible policy γ and given a demand scenario $w(.)$ we are able to build trajectories $x(\cdot)$ for battery and hot water tank: the dynamic is initiate at time t_0 by:

$$x_0 = x_{ini}$$

and the trajectory is computed step by step with following equation:

$$\begin{aligned} u_t &= \gamma(t, x_t) \\ x_{t+1} &= f_t(x_t, u_t, w_t) \end{aligned}$$

We obtain the payoff associated to the demand scenario $w(\cdot)$:

$$J^\gamma\big(w(\cdot)\big) = \sum_{t=0}^{T} L_t(x_t, u_t, w_t) \tag{11}$$

The expected payoff associated with the policy γ is:

$$\mathbb{E}\Big(J^\gamma(w(\cdot))\Big), \tag{12}$$

where the expectation $\mathbb{E}$ is taken with respect to the product probability $\mathbb{P}$. We evaluate (12) by the Monte-Carlo method using N_s scenarios $\Big(w^1(\cdot), \cdots, w^{N_s}(.)\Big)$:

$$\frac{1}{N_s} \sum_{s=1}^{N_s} J^\gamma\big(w^s(\cdot)\big). \tag{13}$$

In the next, we will evaluate each algorithm with this method:

- We compute trajectories with the help of the function γ given by the algorithm.
- We evaluate the average costs with Equation (13).
- We consider the distribution of the costs.

The following code generate scenarios that can be used for simulation:

```
# Read assessment scenarios:
De_assess = readcsv("demandelec_assess.csv")
Dt_assess = readcsv("demandtherm_assess.csv");
scenarios = zeros(model.stageNumber, 30, 2)
scenarios[:, :, 1] = De_assess
scenarios[:, :, 2] = Dt_assess;
```

The data used for scenarios generation can be obtained here in zip format code.zip or tgz code.tgz.

Question 1

1. *Copy this code and execute it.*
2. *Plot 10 scenarios of electrical demand and 10 scenarios of thermal demand.*

2.2 Simulation of a given policy

The function `simulate` is used to simulate a policy:

```
function simulate(scenarios, policy)
    nassess = size(scenarios, 2)
    ntime = size(scenarios, 1)

    x0 = [.6, 20.]
    stocks = zeros(nassess, ntime, 2)
    controls = zeros(nassess, ntime-1, 3)
    costs = zeros(nassess)

    # Set first value of stocks equal to x0:
    for i in 1:nassess
        stocks[i, 1, :] = x0
    end

    for t=1:ntime-1
        for k = 1:nassess
            state_t = stocks[t, k, :]
            aleas = scenarios[t, k, :]

            uc = policy_mpc(t, state_t)

            xf = dynamic(t, state_t, uc, aleas)

            stocks[k, t+1, :] = xf
            controls[k, t, :] = uc
            costs[k] += cost(t, state_t, uc, aleas)
        end
    end
```

```
    return costs, stocks, controls
end
```

Question 2

1. *What is the inputs of the function* `simulate`*? And its outputs?*
2. *Relate this function with the theoritical model above.*

We consider a first policy, given by the echelon-based heuristic:

- If the tank's level is less than a given critical level H^c, then switch on the CHP.
- Once the tank is filled, switch off the CHP.

Question 3

1. *Implement this heuristic. The function should take as input the time t and the current state x_t and return a control u_t.*
2. *Apply this policy along the scenarios generated in Question 1.*
3. *Show the costs' distribution. Plot 10 trajectories for states and controls.*

3 Solving by two different methods

We are going to compare two policies to solve the problem (8):

- a policy produced by the so called Model Predictive Control (MPC),
- a policy based upon Dynamic Programming (DP).

We will assess each policy along the scenarios generated in Question 1. However, these policies need to consider some scenarios as input for their algorithms. To be fair, we will use optimization scenarios distinct from the assessment scenarios considered in Question 1.

As we are considering a real world problem, we do not have that much scenarios recorded. That is why we use only 30 scenarios for optimization and 30 scenarios for assessment.

3.1 Model Predictive Control

As time goes on, MPC computes at the beginning of each period $[\tau, \tau+1[$ the control as a function of the current time and state x_τ. Here are the details:

- Derive a deterministic scenario $\overline{w}(\cdot) = (\overline{D}_\tau^{el}, \overline{D}_\tau^{th}, \ldots, \overline{D}_T^{el}, \overline{D}_T^{th})$ of demands (forecast) with the optimization scenarios.
- Solve the deterministic optimization problem

$$\min_{u_\cdot} \sum_{t=\tau}^{T-1} L_t(x_t, u_t, \overline{w}_t) \tag{14}$$

$$\begin{aligned} s.t. \quad u_\cdot &= (u_\tau, \ldots, u_{T-1}) \\ x_{t+1} &= f_t(x_t, u_t, \overline{w}_t) \\ x^\flat &\leq x_t \leq x^\sharp \\ u &\in \mathbb{U}(x_t) \end{aligned}$$

- Apply only the first control $u_\tau^\sharp$ at time τ, and iterate at time $\tau + 1$

The problem (14) returns a control $u_t^\sharp$ for all time t and state x_t. This problem gives the MPC policy:

$$\begin{array}{rccl} \gamma^{MPC} & : & \mathbb{T} \times \mathbb{X} & \to \mathbb{U} \\ & & (t,\ x_t) & \to u_t^\sharp \end{array} \tag{15}$$

γ^{MPC} is implemented in the function `policy_mpc`:

```
function policy_mpc(t::Int64, x0::Array{Float64})
    tf = NTIME - t

    m = Model()

    # Define constraints other variables:
    @variable(m,  19           <= h[1:(tf+1)]  <= 30)
    @variable(m,  0.5          <= b[1:(tf+1)]  <= 3)
    @variable(m, -DELTA_B_MAX  <= Fb[1:(tf)]   <= DELTA_B_MAX)
    @variable(m,  0.           <= Fa[1:(tf)]   <= 5.5)
    @variable(m,  Fh[1:(tf)]   >=0)
    @variable(m,  0 <= Yt[1:(tf)] <= 1)
    @variable(m,  Z1[1:(tf)] >= 0)


    # Set optimization objective:
    @objective(m, Min, sum{P_BURNER*Fa[i] + P_CHP*Yt[i] +
                           Z1[i] + P_TH*Fh[i], i = 1:tf})

    for i in 1:tf
        @constraint(m, b[i+1] - ALPHA_B * b[i] +
                       DT*BETA_B*Fb[i] == 0)
        @constraint(m, h[i+1] - ALPHA_H * h[i] -
                       DT*BETA_H*(Yt[i]*P_CHP_THERM + Fa[i] -
                       previsions[(i+t -1), 2] + Fh[i]) == 0)

        @constraint(m, Z1[i] >=  (previsions[(i+t -1), 1] -
                                  Fb[i] - Yt[i]*P_CHP_ELEC)*priceElec[t+i-1])
        @constraint(m, Z1[i] >= -(previsions[(i+t -1), 1] -
                                  Fb[i] - Yt[i]*P_CHP_ELEC)*P_INJECTION)
    end
```

```
    # Add initial constraints:
    @constraint(m, b[1] == x0[1])
    @constraint(m, h[1] == x0[2])

    status = solve(m)
    return [getvalue(Yt)[1]; getvalue(Fb)[1]; getvalue(Fa)[1]]
end
```

Question 4

1. *Explain what the above code is doing.*
2. *Apply MPC policy along the scenarios generated in Question 1.*
3. *Show the costs' distribution. Plot 10 trajectories for states and controls.*
4. *Does the trajectory of the battery's state of charge seem reasonable?*

3.2 Stochastic Dynamic Programming

Dynamic programming works in two stages: an *offline* stage where value functions are computed, and an *online* stage where these functions are used to compute a policy.

Offline: computation of value functions

- Use optimization scenarios to estimate expectation $\widehat{\mathbb{E}}$
- Use these marginal distributions to compute so-called value functions, given by

$$V_t(x_t) = \min_{u_t \in \mathbb{U}} \mathbb{E}\big[L_t(x_t, u_t, w_t) + V_{t+1}\big(f_t(x_t, u_t, w_t)\big)\big]$$

Online: construction of the policy DP policy computes at the beginning of each period $[\tau, \tau+1[$ the control as a function of the current time τ and state x_τ:

- Solve the optimization problem with the help of the previously computed value functions $(V_t)_{t=0}^{T_f}$.

$$u_\tau^\sharp \in \underset{u_\tau \in \mathbb{U}}{\arg\min}\ \mathbb{E}\big[L_t(x_\tau, u_\tau, w_\tau) + V_{\tau+1}\big(f_t(x_\tau, u_\tau, w_\tau)\big)\big]$$

- Apply control $u_\tau^\sharp$ at time τ, and iterate at time $\tau+1$.

$x_{t+1} = f_t(x_t, u_t, w_t)$ is a functionnal relation, lying in the continuous state space $\mathbb{X}$. As we use a computer we will discretize state and control spaces to implement dynamic programming. We restrict ourself to a subset $\mathbb{X}^d \subset \mathbb{X}$ for states and $\mathbb{U}^d \subset \mathbb{U}$ for controls.

Implementation The Dynamic Programming solver is already implemented. It is called by the function:

```
Vs = solve_DP(sdpmodel, paramSDP, 1)
```

which return the Bellman functions in a multidimensional array `Vs`.

Question 5 *Generate marginal laws with the help of the given optimization scenarios and the function* `generate_laws`*.*

Question 6

1. *Compute Bellman functions with the function* `solve_DP`*.*
2. *Apply Dynamic Programming policy along scenarios generated in Question 1.*
3. *Show the costs' distribution. Plot 10 trajectories for states and controls.*

Question 7 *Plot different Bellman functions with the help of the function* `plot_bellman_functions`*. Comment the evolution of the shape of these functions along time.*

Question 8

1. *Find the discretization used for states and controls. What is the cardinal of the set* $\mathbb{X}^d$ *(approximation of the space* $\mathbb{X}$*)? And those of the set* $\mathbb{U}^d$*?*
2. *What happens if we refine the discretization of tank and battery by 10?*

3.3 Comparison between different policies

We want to compare three different features for each algorithm:

- the offline computation time,
- the online computation time,
- the average costs obtained upon the assessment scenarios.

Policy	Offline computation time	Online computation time	Average Costs
Heuristic			
MPC			
DP			

Table 2: Benchmark

Question 9 *Fill the Table 2 with the results previously obtained.*

A Discretization and interpolation schemes

We have two stocks: a battery and thermal tank. Even if the dynamic programming principle still holds, the numerical implementation is not as straightforward as in the unidimensional case.

A.1 Discretization

We need to discretize state and control spaces to apply dynamic programming. We restrict ourself to a subset $\mathbb{X}^d \subset \mathbb{X}$ for states and $\mathbb{U}^d \subset \mathbb{U}$ for controls. The policy find by dynamic programming is then suboptimal.

To iterate upon different states in $\mathbb{X}^d$, we define a bijective mapping between $\mathbb{X}^d$ and $\mathbb{N}$. This mapping is written:

$$\phi : \mathbb{X}^d \to \mathbb{N} \tag{16}$$

Question 10

1. *Find a mapping that satisfy to the conditions previously stated.*
2. *Implement it.*

A.2 Interpolation

With this discretization, we know the value of Bellman function V only upon the grid $\mathbb{X}^d$, and not upon the whole state $\mathbb{X}$.

We recall that the next state is given by the state equation: However, even if $x_t \in \mathbb{X}^d$ and $u_t \in \mathbb{U}^d$, we can not ensure that $x_{t+1} \in \mathbb{X}^d$ to compute future cost $V_{t+1}(x_{t+1})$. That is why we need to use an interpolation scheme to interpolate $V_{t+1}(x_{t+1})$ with the values of V_{t+1} in $\mathbb{X}^d$.

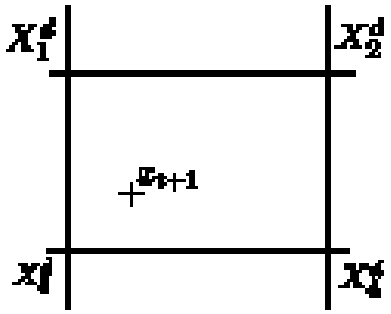

Figure 4: x_{t+1} could be outside the grid defined by $\mathbb{X}^d$

Question 11 *With the notation introduced in Figure 4, write the linear interpolation of x_{t+1} w.r.t X_i^d.*