ME-425: Mini-Project
Building

Due on Thursday, June 9, 2016

Colin Jones

Olivier Lévêque & Maxime Maurin

Contents

1 Introduction **3**
1.1 Files . . . 3

2 Getting acquainted - Building model **4**

3 First MPC controller **5**
3.1 Choice and influence of the tuning parameters on the MPC scheme . . . 5
3.2 Codes and results . . . 5

4 Economic MPC and Soft Constraints **8**
4.1 Motivation and choice of the cost function to penalize the slack variables . . . 8
4.2 Codes and results . . . 8

5 Variable Cost **11**

6 Night Setbacks **12**

7 Battery Storage **13**
7.1 Results and battery usage . . . 13
7.2 Influence of the storage capacity . . . 13

1 Introduction

In this project we have implemented an MPC controller for a building's heating system. We have designed an MPC controller to minimize the total cost of operation that takes into account the prediction of the weather, and studied the impact of additional heat storage.

1.1 Files

For easier understanding our code, you can run our project from `main.m` and read the code part corresponding to each section with:

- `trackingMPC.m` (for section 1),
- `economicMPC_sc.m` (for section 2),
- `economicMPC_sc_vc.m` (for section 3),
- `economicMPC_sc_vc_sb.m` (for section 4),
- `economicMPC_sc_vc_sb_battery.m` (for section 5).

The following tree illustrates the organisation of MATLAB repertory submit with this report.

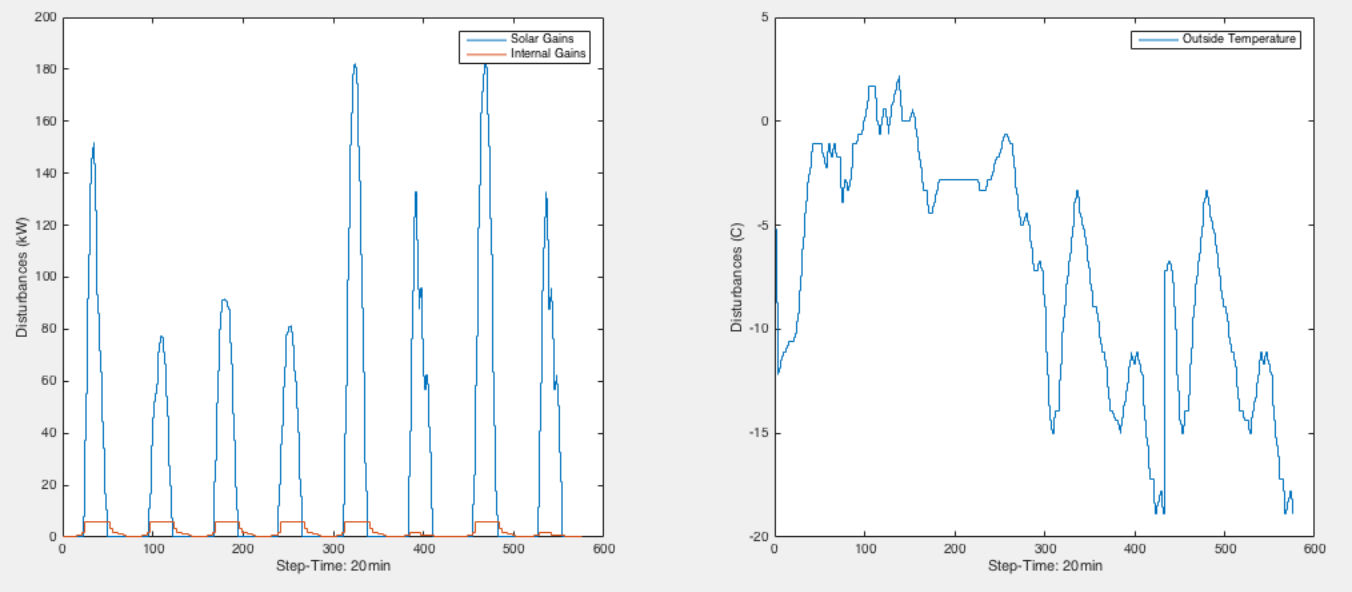
MATLAB
- `battery.mat`
- `building.mat`
- `main.m` (modified)
  - ⟶ `trackingMPC.m` (created)
  - ⟶ `economicMPC_sc.m` (created)
  - ⟶ `economicMPC_sc_vc.m` (created)
  - ⟶ `economicMPC_sc_vc_sb.m` (created)
  - ⟶ `economicMPC_sc_vc_sb_battery.m` (created)
- `simBuildStorage.m` (modified)
- `simBuild.m` (modified)
  - ⟶ `shiftPred.m` (modified)
- `tuning_horizon_length.m` (created)
- `battery_usage.m` (created)
- FIGURES

Organisation of MATLAB repertory

This small report presentes you plots and answers to questions posed in the subject avaibled on moodle: `http://moodle.epfl.ch/mod/resource/view.php?id=922124`.

2 Getting acquainted - Building model

We have ploted, on the same frame, the three disturbance inputs.



Three disturbance inputs

The first figure (on the left) illustrates the power repartition of the Solar Gains and Internal Gains during height days (576x20min), and the second one (on the right) represents the outside temperature over the same period.

We can notice on the first plot that the behavior of curves is cyclic. In fact, as the figure shows the disturbances over height days, we can observe positive solar gains during the day and nul solar gains during the night. So naturally, the internal gains increase during the day to reach a maximum and decrease during the night.

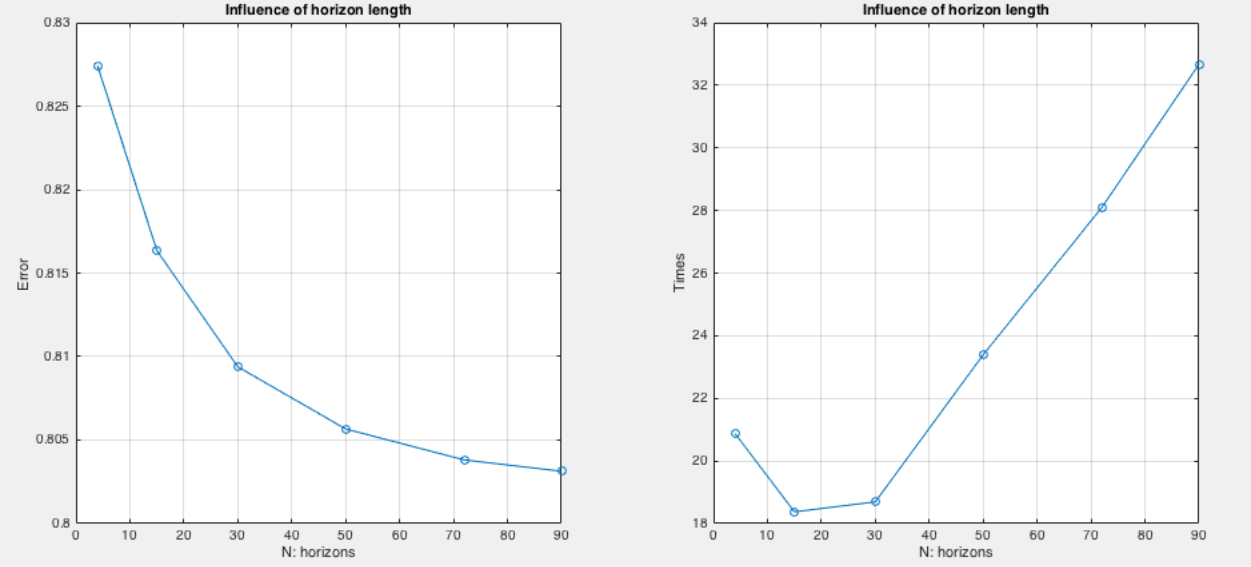
It is more complicated to explain the behavior of the outside temperature which depends of external conditions. We can just say that the outside temperature change over the time.

3 First MPC controller

3.1 Choice and influence of the tuning parameters on the MPC scheme

The regulation system of the temperature in buildings is a stable system using a long horizon. Hence feasibility and stability are achieved in a large neighborhood of the origin. Moreover, a controller must provide some input in every circumstance. So, we decided not to use terminal sets and terminal constraints.

Concerning the choice of MPC controller horizon parameter N, we decided to choose $N = 72$, which corresponds to a MPC horizon of one day. In fact, we have noted that the largest absolute error on the output temperature, when compared to the reference one, varies insignificantly when N increases starting from a value of N around 72. The following figure illustrates this. Choosing a longer N isn't then necessary since we wouldn't gain in accuracy but we would have a longer computation time.



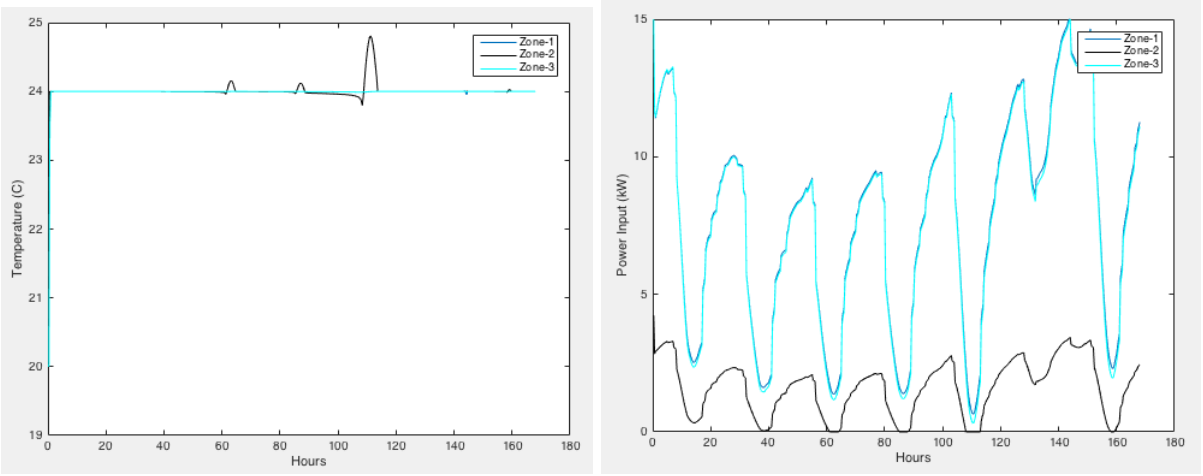
Influence of the horizon length

In consequence, T, the number of time steps of the simulation, has to be equal to the number of time steps of the given disturbances minus $N - 1$.

For the choice of R, we decided to make it an identity matrix. In fact the value of R doesn't have any influence on the minimization problem since the objective function depends only on the output values.

3.2 Codes and results

The following figures illustrate the results of our first MPC controller.



Results of tracking MPC

We can observe with the first figure (on the left) that the reference outpout values ($24C$) for each zone are achieved most of the time, and that the constraints of maximum and minimum temperatures are not violated.
We can notice too in the second figure (on the rigth) the cyclical behavior of the power input. Indeed, the simulation was made over few days.
We can also notice that zone 2 gets warmer than the other zones (in the first figure), and that's why its power input is significantly less than for the other zones (in the second figure).

The following Matlab code was used to define and simulate our first MPC controller.

Listing 1: trackingMPC.m

```
%%% Olivier Leveque & Maxime Maurin --  7 June 2016
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xt, yt, ut, t] = trackingMPC(ssM, param_constraints, param_objective, ...
    param_simulation)

%parameters of the Building Model
A = ssM.A;
Bu = ssM.Bu;
Bd = ssM.Bd;
C = ssM.C;

%other parameters
nx = length(A); %number of states
ny = size(C,1); %number of outputs
nu = size(Bu,2); %number of inputs
nd = size(Bd,2); %number of disturbances

%define simulation parameters
N = param_simulation.N;
T = param_simulation.T;

%define objective parameters
R = param_objective.R;
yref = param_objective.yref;

%define constraints matrix
Mu = param_constraints.Mu;
```

```
My = param_constraints.My;
mu = param_constraints.mu;
my = param_constraints.my;

%define optimization variables
x = sdpvar(nx, N, 'full'); %states
x0 = sdpvar(nx, 1, 'full'); %initial states
y = sdpvar(ny, N, 'full'); %outputs
u = sdpvar(nu, N, 'full'); %inputs
d = sdpvar(nd, N, 'full'); %disturbances

%define constraints and objective
constraints = [];
objective = 0;
for i = 1:N-1
    if i == 1
        constraints = [constraints, x(:,i+1) == A*x0 + Bu*u(:,i) + Bd*d(:,i)]; ...
            %system dynamics - first step
    else
        constraints = [constraints, x(:,i+1) == A*x(:,i) + Bu*u(:,i) + Bd*d(:,i)]; ...
            %system dynamics
    end
    constraints = [constraints, y(:,i) == C*x(:,i)]; %observability
    constraints = [constraints, Mu*u(:,i) ≤ mu]; %input constraints
    constraints = [constraints, My*y(:,i) ≤ my]; %output constraints
    objective = objective + (y(:,i)-yref)'*R*(y(:,i)-yref); %cost function
end
constraints = [constraints, y(:,N) == C*x(:,N)]; %observability
constraints = [constraints, Mu*u(:,N) ≤ mu]; %terminal input constraints
constraints = [constraints, My*y(:,N) ≤ my]; %terminal output constraints
objective = objective + (y(:,N)-yref)'*R*(y(:,N)-yref); %terminal cost

%create a controller
ops = sdpsettings('verbose', 1);
parameters_in = [x0; d(:)];
solutions_out = u;
controller = optimizer(constraints, objective, ops, parameters_in, solutions_out);

%simulate the system
option = 1; %simulation with no night-setbacks and no variable cost
[xt, yt, ut, t, ¬] = simBuild(controller, T, @shiftPred, N, option);

end
```

4 Economic MPC and Soft Constraints

4.1 Motivation and choice of the cost function to penalize the slack variables

The cost function considers the true cost of operation and a penalty on the slack variables, used for soft constraints.
The objective function is written:

$$\begin{aligned} J &= \sum_{i=1}^{N} ([ccc] * u(i) + \epsilon_i^T * S * \epsilon_i) \\ &= \sum_{i=1}^{N} (c * \sum_{k=1}^{3} u_k(i) + \epsilon_i^T * S * \epsilon_i) \end{aligned}$$

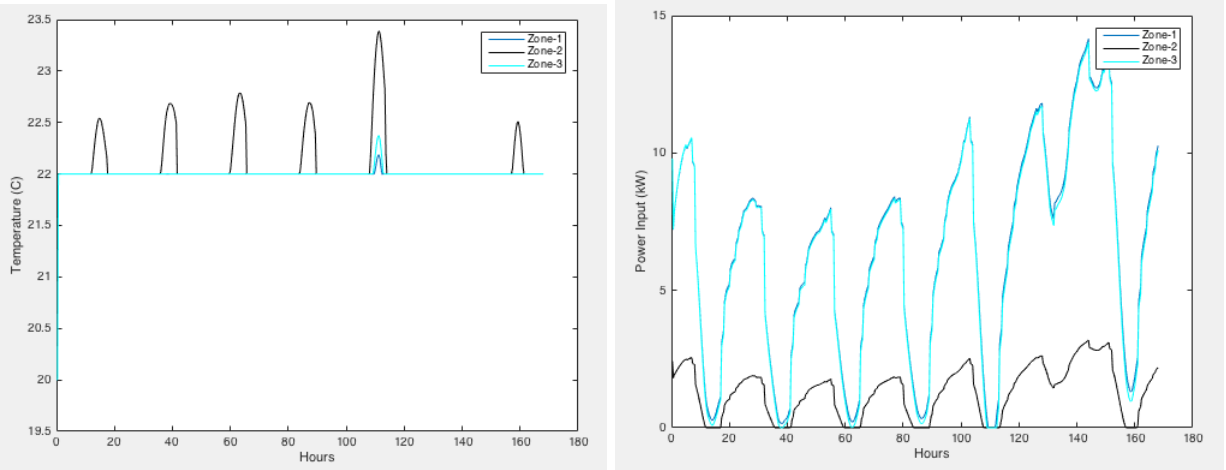
With this new cost function, we can implement an economic MPC, and minimize the true cost. This true cost is given by the sum, along the simulation time, of the power inputs multiplied by the price of electricity. We have implemented it with Matlab :

```
objective = objective + c*sum(u(:,i))+ epsilon(:,i)'*S*epsilon(:,i);
```

However the state constraints may lead to infeasibility, and the controller must provide some input in every circumstance, so we need to add soft constraints to the objective function. In the objective function that we defined, we chose a quadratic penalty. The choice of the value of S is a tradeoff between the minimization of the duration of violations regarding the input, and the minimization of the size of the violation. We found that a value of $S = 50$ is suitable for our case.

4.2 Codes and results

The following figures illustrate the results of our economic MPC controller using soft constraints and a fixed cost.



Results of economic MPC with soft contraints and a fixed cost

We can see in the first figure (on the left) that the temperatures tend to be at $22C$, which is the cheapest option to achieve. The second figure (on the rigth) clearly shows that we use less power input in comparison to the previous MPC controller.

The following Matlab code was used to define and simulate our economic MPC controller.

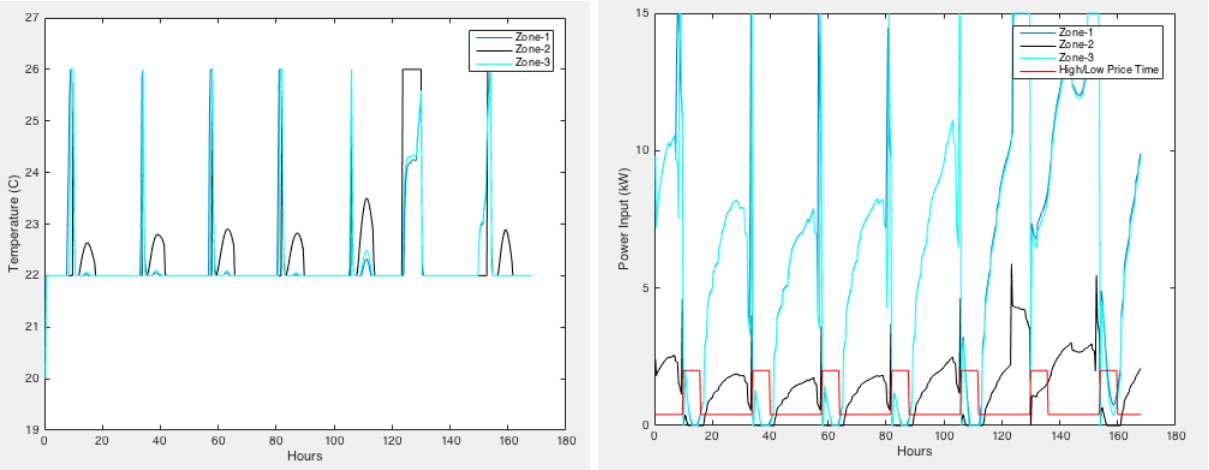
Listing 2: economicMPC_sc.m

```
%%% Olivier Leveque & Maxime Maurin --  7 June 2016
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xt, yt, ut, t, total_cost] = economicMPC_sc(ssM, param_constraints, ...
    param_objective, param_simulation)

%parameters of the Building Model
A = ssM.A;
Bu = ssM.Bu;
Bd = ssM.Bd;
C = ssM.C;

%other parameters
nx = length(A); %number of states
ny = size(C,1); %number of outputs
nu = size(Bu,2); %number of inputs
nd = size(Bd,2); %number of disturbances

%define simulation parameters
N = param_simulation.N;
T = param_simulation.T;

%define objective parameters
R = param_objective.R;
yref = param_objective.yref;
c = param_objective.c;
S = param_objective.S;

%define constraints matrix
Mu = param_constraints.Mu;
My = param_constraints.My;
mu = param_constraints.mu;
my = param_constraints.my;

%define optimization variables
x = sdpvar(nx, N, 'full'); %states
x0 = sdpvar(nx, 1, 'full'); %initial states
y = sdpvar(ny, N, 'full'); %outputs
u = sdpvar(nu, N, 'full'); %inputs
d = sdpvar(nd, N, 'full'); %disturbances
epsilon = sdpvar(ny, N, 'full'); %slack variables

%define constraints and objective
constraints = [];
objective = 0;
for i = 1:N-1
    if i == 1
        constraints = [constraints, x(:,i+1) == A*x0 + Bu*u(:,i) + Bd*d(:,i)]; ...
            %system dynamics - first step
    else
        constraints = [constraints, x(:,i+1) == A*x(:,i) + Bu*u(:,i) + Bd*d(:,i)]; ...
            %system dynamics
    end
    constraints = [constraints, y(:,i) == C*x(:,i)]; %observability
    constraints = [constraints, Mu*u(:,i) <= mu]; %input constraints
    constraints = [constraints, My*y(:,i) <= my + kron([1; -1],epsilon(:,i))]; %soft ...
        output constraints
    constraints = [constraints, zeros(ny,1) <= epsilon(:,i)]; %slack variable constraints
    objective = objective + c*sum(u(:,i)) + epsilon(:,i)'*S*epsilon(:,i); %economic ...
        cost function
```

```
end
constraints = [constraints, y(:,N) == C*x(:,N)]; %observability
constraints = [constraints, Mu*u(:,N) ≤ mu]; %terminal input constraints
constraints = [constraints, My*y(:,N) ≤ my + kron([1; -1], epsilon(:,N))]; %terminal ...
    soft output constraints
constraints = [constraints, zeros(ny,1) ≤ epsilon(:,N)]; %terminal slack variable ...
    constraints
objective = objective + c*sum(u(:,N)) + epsilon(:,N)'*S*epsilon(:,N); %terminal ...
    economic cost function

%create a controller
ops = sdpsettings('verbose', 1);
parameters_in = [x0; d(:)];
solutions_out = u;
controller = optimizer(constraints, objective, ops, parameters_in, solutions_out);

%simulate the system
option = 1; %simulation with no night-setbacks and no variable cost
[xt, yt, ut, t, ¬] = simBuild(controller, T, @shiftPred, N, option);

%compute total economic cost
total_cost = c*sum(ut(:));

end
```

5 Variable Cost

The following figures illustrate the results of our economic MPC controller using soft constraints and a variable cost.



Results of economic MPC with soft contraints and a variable cost

We can see in the second figure (on the rigth) that the power is consumed massively when the electricity is cheap, and less when it is expansive. We can also notice that 2 hours before the electricity becomes expansive, the controller decides to buy power massively so that the building rather uses this acquired thermal inertia than a bought power at a higher price.

In consequence, the temperature of each zone increase significantly before the electricity becomes expansive as the fist figure (on the left) illustrates it. However, we notice that the thermal intertia of the building isn't important because the temperatures decrease quickly afterwards.

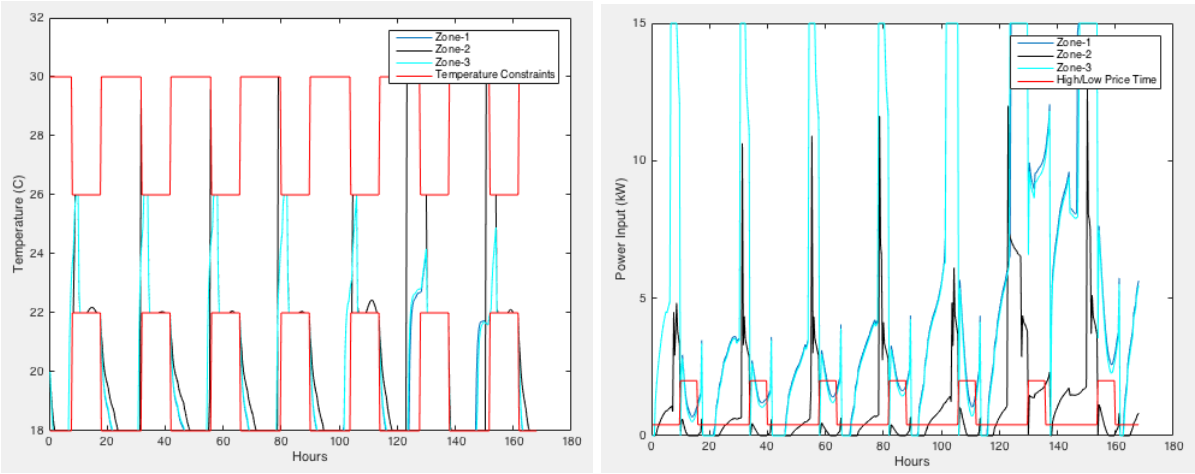
We have calculated the total cost of electricity spent for both cases (for all zones):

- with a fixed cost, the system spends $1400;
- with a variable cost, the system spends $356.

So, it makes sense to use this controller rather than the previous to save money.

6 Night Setbacks

The following figures illustrate the results of our economic MPC controller using soft constraints, a variable cost and nights setbacks.



Results of economic MPC with soft contraints, a variable cost and night setbacks

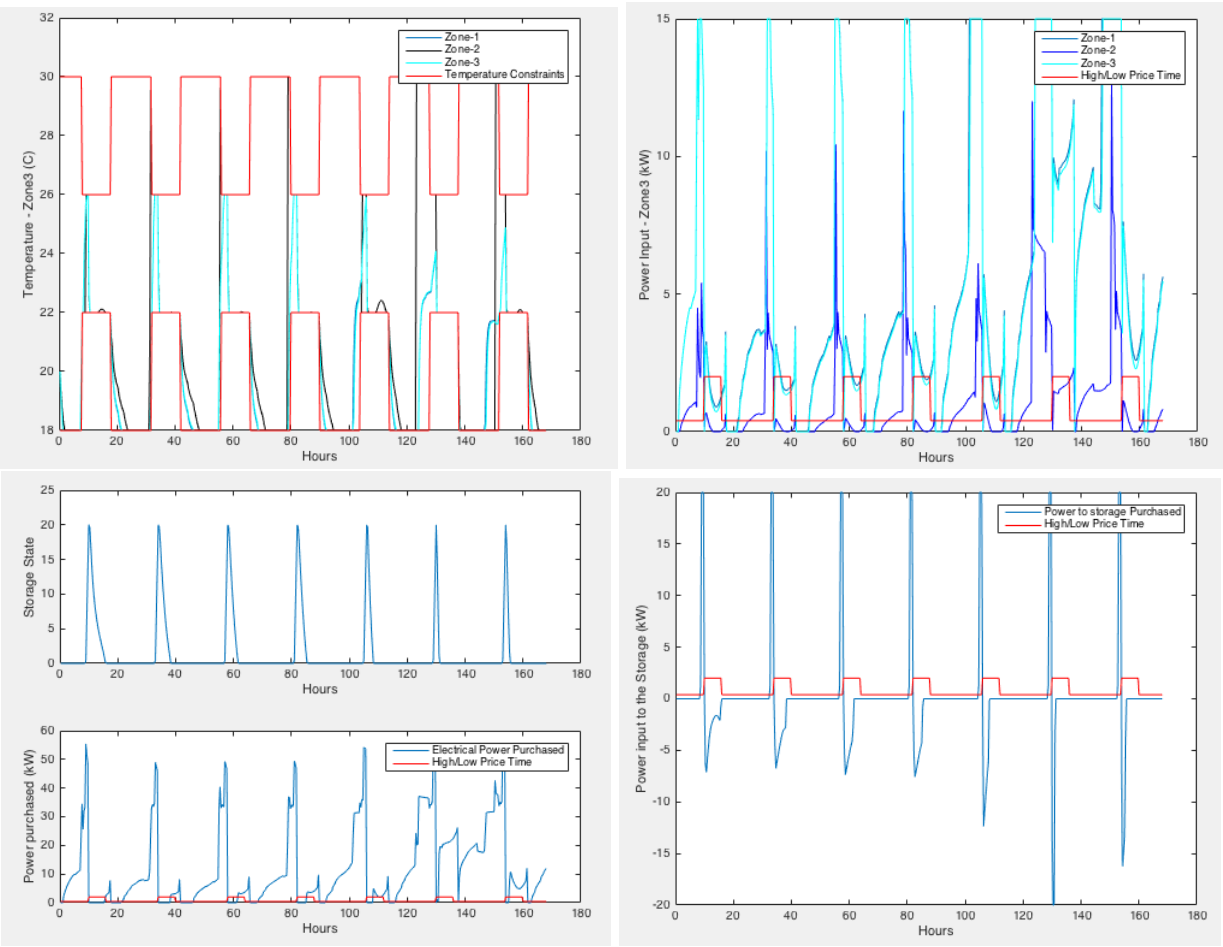
The first figure (on the left) shows that the night setbacks allows the temperatures of each zone to vary smoother than in the previous simulation. In fact, the thermal inertia of the building is more used because now the controller is able to increase the power input, just before that the cost increases. Unlike the previous simulation where most of the power was acquired during nights, the controller prefers now waitting longer to comsume the power from the grid (second figure on the rigth).
This best way, to use the thermal inertia of the building, encourage to consume power during the day in comparaison with the previous simulation, so the total cost is higher than previously.

The final cost reaches $365 with the night setbacks. However, this growth isn't so important and makes sense to use the night setbacks.

7 Battery Storage

7.1 Results and battery usage

The following figures illustrate the results of our economic MPC controller using soft constraints, a variable cost, nights setbacks and battery storage.



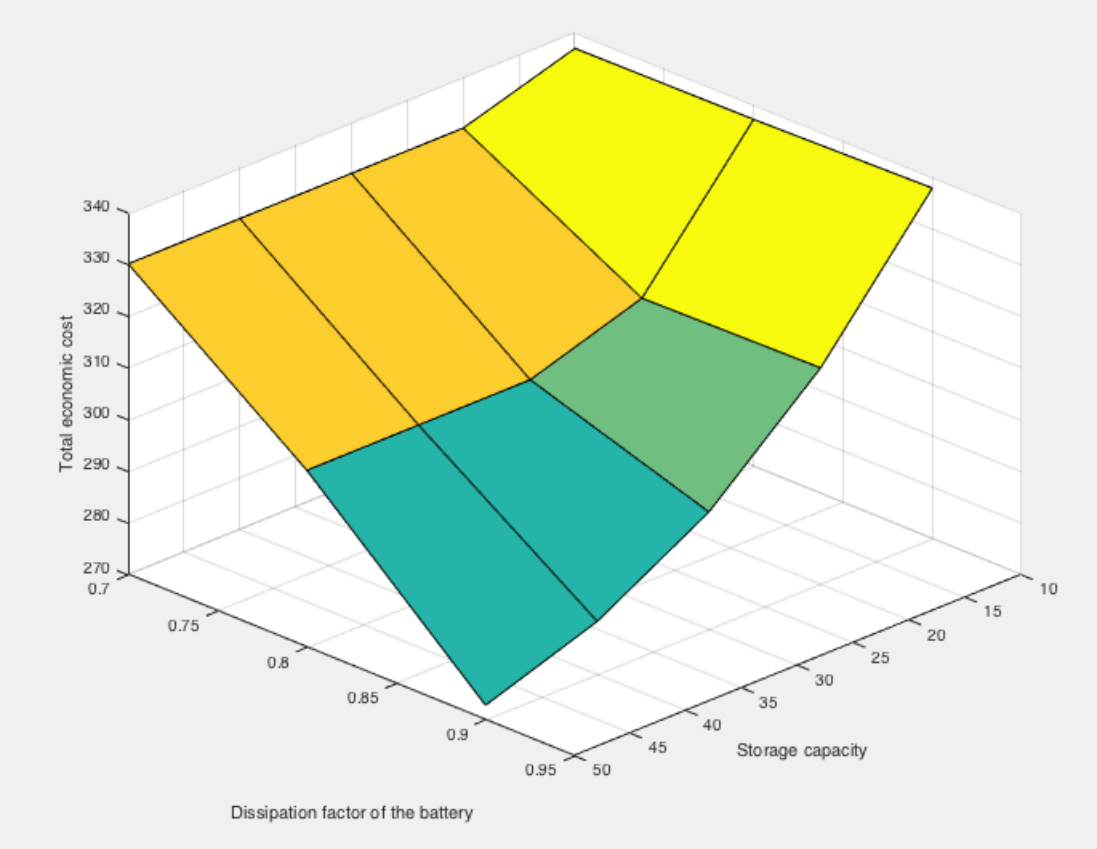
Results of economic MPC with soft contraints, a variable cost, night setbacks and battery storage

The battery is charged during night and discharged during the day. Adding a battery to the building permits to buy electricity at a low price during night, and consumes this electricity during the following day instead of consuming electricity at a high price.

The total cost of bought electricity is now \$312 compared to \$365 without batteries.

7.2 Influence of the storage capacity

The following figure illustrates the influence of storage capacity and dissipation factor on the total cost.



Cost of total input power for economic MPC with soft contraints, a variable cost, night setbacks and battery storage with different storage capacities and dissipation factors

We have decided to observe the impact of the storage capacity $\tilde{x}_{max}$ and the dissipation factor α of the battery on the total energy consumption.
So we have displayed this figure which illustrates the influence of $\tilde{x}_{max}$ and α on the total cost. We chose $\tilde{x}_{max}$ from 10 to 50 kWh (10, 20, 30, 40, 50), and α from 0.7 to 0.9 (0.7, 0.8, 0.9).
(In the previous simulations $\tilde{x}_{max} = 20$ kw$h$ and $\alpha = 0.98$).

Thanks to different figures displayed by our Matlab code but didn't print in this report because there were so many, we can notice how the battery can be used for a longer time and at a higher state of charge when less dissipation occurs.

- Having more capacity makes the controller charge the battery more during night, so less electricity is bought with a high price.
- Increasing both the capacity and the dissipation factor (having less dissipation), energy savings are made over the total period.