Félix MARTEL
Oussama ZOUAGHIA

Mini-Project
Building Climate Control

Getting acquainted – Building model

1. **Plot the three disturbance inputs and explain anything noticeable**

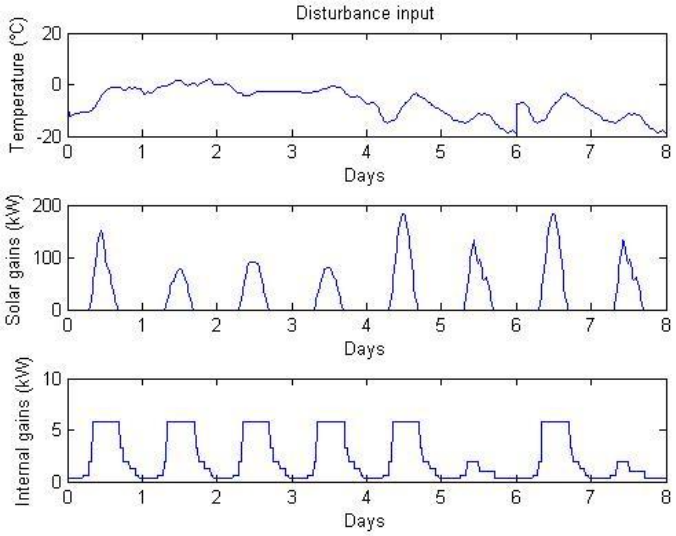

Figure 1: The three disturbance inputs

We notice that:

- The temperature is below 0°C during all the simulation
- The solar gain has a cyclic shape (1-day periodic), which is normal
- The internal gain (the thermal power in each zone of the building due to occupancy, lighting, electrical equipment, etc.) has a 1-day period too and it saturates during the peak of the office hour when everyone is in the office

First MPC controller

After discussing it with the assistants, we decided to do not use a terminal set for this problem. Because, for building control applications usually the objective is to not just to stabilize the room temperatures to a specific value but to keep the temperature within certain comfort range, while minimizing the operational costs. In this setting, the traditional terminal set is not very useful, since we are not trying to stabilize the system to its steady state. Moreover, for large systems it is very difficult to compute the terminal set, so often it is not used in practice.

1. **Explanation of the influence of the tuning parameters on the MPC scheme and choice of your tuning parameters**

One of the tuning parameters is the length of the horizon. Increasing it leads to more robust results since it gets closer to the infinite-horizon case (the perfect optimization). However, by doing so, we increase the calculation time. In our case, we chosed N = 72 so that we take into account one day (20min*72 = 24h). One another parameter is R and in our case we set it to 1 (it is the only one that intervenes in our case). Tuning it would be interesting if we had a terminal cost, which is not the case.

2. Code generating the plots and comments on the plots

```
% Setting the parameters
N = 72; % Horizon
yRef = [24 24 24]'; % Reference
R = 1;
T = 10; % Simulation length in time-steps

% Defining the input and output constraints
      ku = [15; 0];
      Hu = [1; -1];

      % Define optimization variables for MPC
      xMPC = sdpvar(10, N,'full');
      uMPC = sdpvar(3, N-1,'full');
      yMPC = sdpvar(3, N,'full');
      d = sdpvar(3, N,'full');
      x0 = sdpvar(10,1,'full'); % Initial State

      % Define the cost function for MPC
      objective = 0;
      for i = 1:N
          objective = objective + (yMPC(:,i)-yRef)'*R*(yMPC(:,i)-yRef);
      end

      % Define constraints
      constraints = [];
      constraints = [constraints, xMPC(:,1) == x0];
      for i = 1:N-1
          % System dynamics
          constraints = [constraints, yMPC(:,i) == C*xMPC(:,i)];
          constraints = [constraints, xMPC(:,i+1) == A*xMPC(:,i) +
Bu*uMPC(:,i) + Bd*d(:,i)];
          % Input constraints
          for j = 1:3
              constraints = [constraints, Hu*uMPC(j,i) <= ku];
          end
      end
      constraints = [constraints, yMPC(:,N) == C*xMPC(:,N)];

      ops = sdpsettings('verbose',1);
      controller = optimizer(constraints,objective,ops,[x0;d(:)],uMPC);
      [xt, yt, ut, t] = simBuild(controller, T, @shiftPred, N, 1);
```

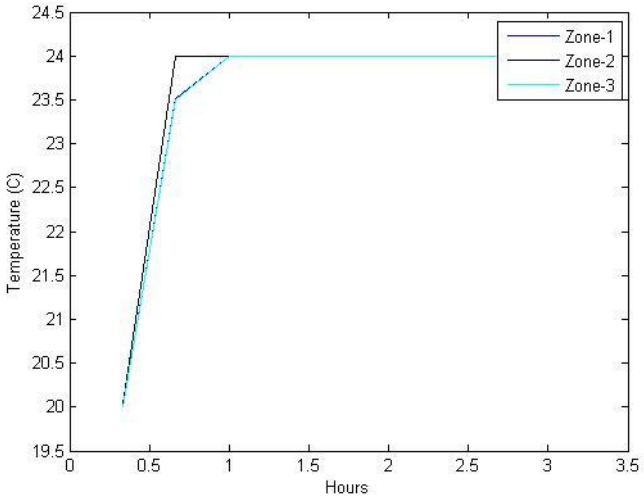

Figure 2: The temperature _ First MPC Controller

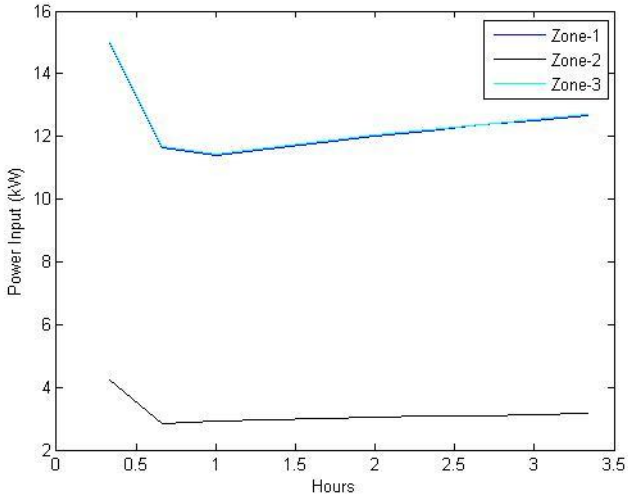

Figure 3: The inputs _ First MPC Controller

We notice that the temperature in the 3 zones converges to 24°C and this convergence is the fastest in the case of the zone 2 although its lower inputs. However, this can be explained by the fact that it is located between the other zones. Moreover, zone 1 and zone 3 has similar inputs and temperature curve.

Economic MPC - Soft Constraints

1. **Choice of the cost function to penalize the slack variables and motivation for it**

The cost function is:

$$obj = \sum_{i=1}^{N} \left(\sum_{j=1}^{3} p.u_{i,j} \right) + eps'.S.eps$$

Where p is the fixed electricity price, $u_{i,j}$ is the input of the zone j at the instant I, eps is the slack variables and S is the cost that corresponds to the slack variables.

We have put the S equal to 10, in order that the slack variables is more penalized than the price of electricity. As consequence, the comfort constraints stay always satisfied (Except in the beginning where the temperature is equal to 20).

2. **Code generating the plots and comments on the plots**

```
% Setting the parameters
N = 72; % Horizon
p = 0.2; % Price of electricity in $/kWh
pM = [p p p]; % Price matrix
yRef = [24 24 24]'; % Reference
S = 10;
T = 300; % Simulation length in time-steps
```

```
11.       % Defining the input and output constraints
12.       ku = [15; 0];
13.       Hu = [1; -1];
14.       ky = [26; -22];
15.       Hy = [1; -1];
16.
17.       % Define optimization variables for MPC
18.       xMPC = sdpvar(10, N,'full');
19.       uMPC = sdpvar(3, N-1,'full');
20.       yMPC = sdpvar(3, N,'full');
21.       epsMPC = sdpvar(3, N,'full'); % Slack variable
22.       d = sdpvar(3, N,'full');
23.       x0 = sdpvar(10,1,'full'); % Initial State
24.
25.       % Define the cost function for MPC: cost = sum(p*input)
26.       objective = 0;
27.       % *) minimize the cost (linked to the price)
28.       for i = 1:N-1
29.           objective = objective + pM*uMPC(:,i);
30.       end
31.       % *) minimize the violation over the horizon
32.       for i = 1:N
33.           objective = objective + epsMPC(:,i)'*S*epsMPC(:,i);
34.       end
35.
36.       % Define constraints
37.       constraints = [];
38.       constraints = [constraints, xMPC(:,1) == x0];
39.       for i = 1:N-1
40.           % System dynamics
41.           constraints = [constraints, yMPC(:,i) == C*xMPC(:,i)];
42.           constraints = [constraints, xMPC(:,i+1) == A*xMPC(:,i) +
    Bu*uMPC(:,i) + Bd*d(:,i)];
43.           % Input constraints
44.           for j = 1:3
45.               constraints = [constraints, Hu*uMPC(j,i) <= ku];
46.           end
47.           % Output constraints
48.           for j = 1:3
49.               constraints = [constraints, Hy*yMPC(j,i) <= ky +
    epsMPC(j,i)];
50.           end
51.       end
52.       % last Output constraint
53.       for j = 1:3
54.           constraints = [constraints, Hy*yMPC(j,N) <= ky + epsMPC(j,N)];
55.       end
56.       constraints = [constraints, yMPC(:,N) == C*xMPC(:,N)];
57.
58.       ops = sdpsettings('solver','gurobi');
59.       controller = optimizer(constraints,objective,ops,[x0;d(:)],uMPC);
60.       [xt, yt, ut, t] = simBuild(controller, T, @shiftPred, N, 1);
```

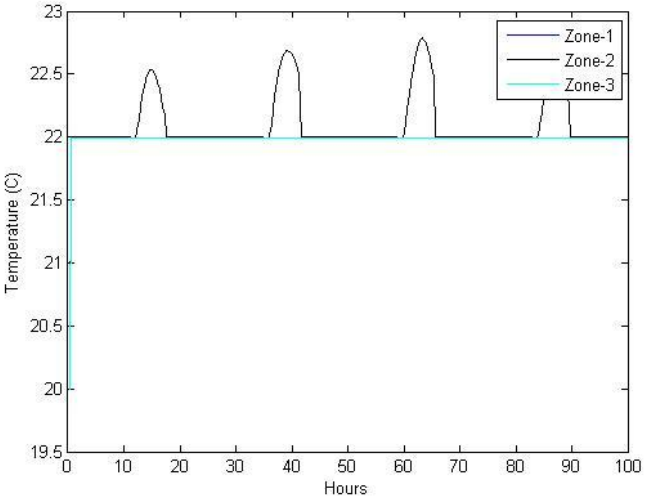

Figure 5: The temperature _ Economic MPC-Soft constraints

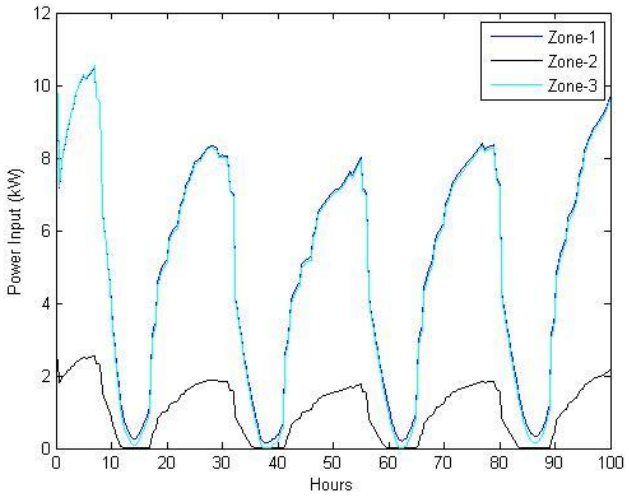

Figure 4: The inputs _ Economic MPC-Soft constraints

In these plots, we can see that the temperature stays in the comfort zone. Similarly to the previous question, the zone 1 and 2 have the same input and the same temperature curve. We can see also that the temperature curve has a tendency to stay at 22 (the minimum of the comfort zone) in order to minimize the electricity cost.

Variable Cost

1. **plots of the response starting from the given initial condition and its explanation**

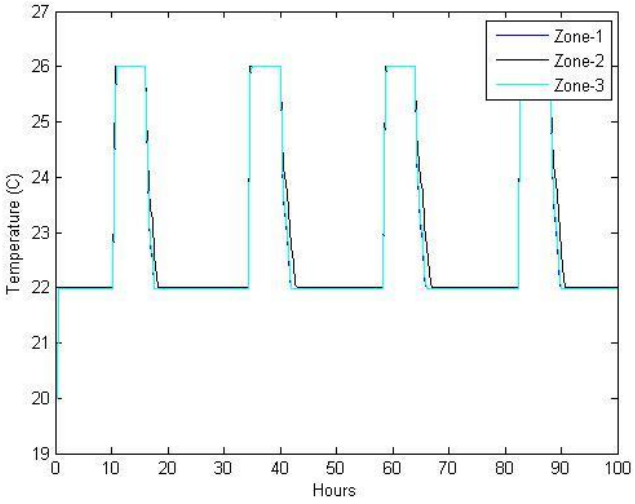

Figure 6: The temperature _ Variable cost

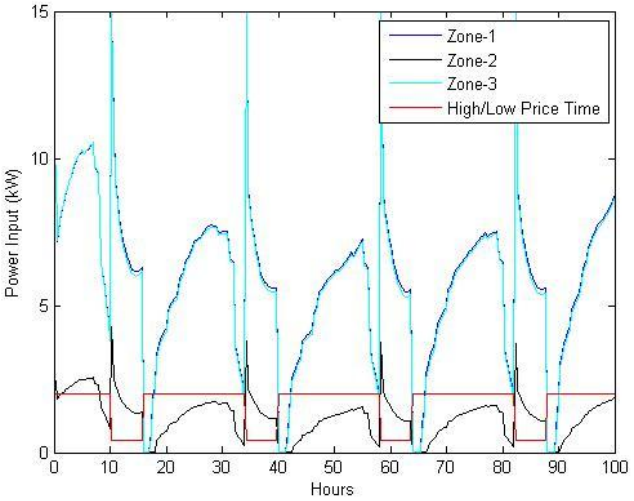

Figure 7: The inputs _ Variable cost

We can see that the climate control has a tendency to increase the temperature until the maximum of the comfort zone when the electricity cost is the lowest. Hence, we minimize the total cost.

Night Setbacks

1. **plots of the response starting from the given initial condition and explanation of the results**

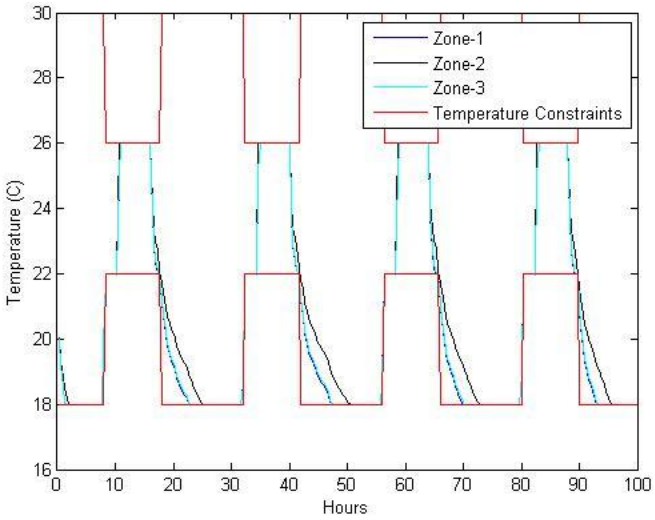

Figure 8: The temperature _ Night Setbacks

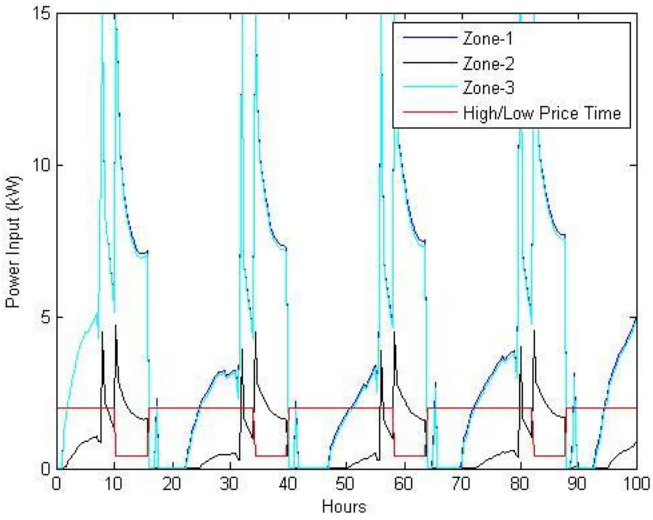

Figure 9: The inputs _ Night Setbacks

We can see that during the night the temperature decreases to the minimum corresponding to the night (18°C). Then, just before the day, the temperature increases to fit the new comfort zone.
Added to this, we can see as in the previous question, the climate control has a tendency to increase the temperature until the maximum of the comfort zone when the electricity cost is the lowest.

Battery Storage

1. **plots of the response starting from the given initial condition.**

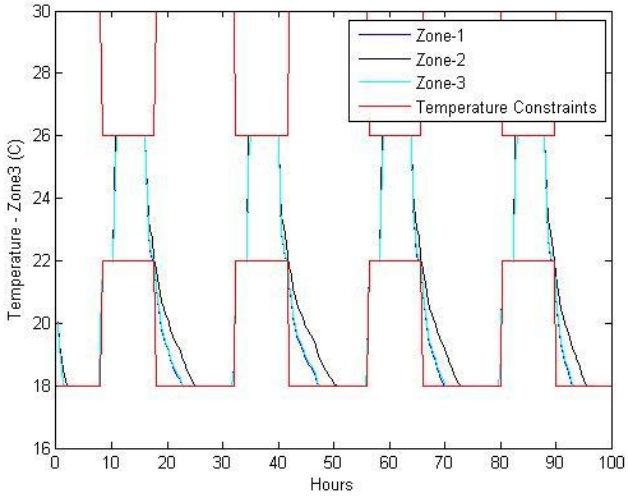

Figure 11: The temperature _ Battery storage

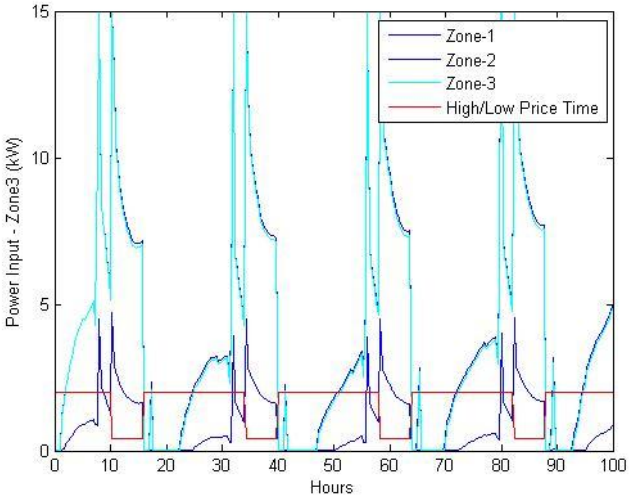

Figure 10: The inputs _ Battery storage

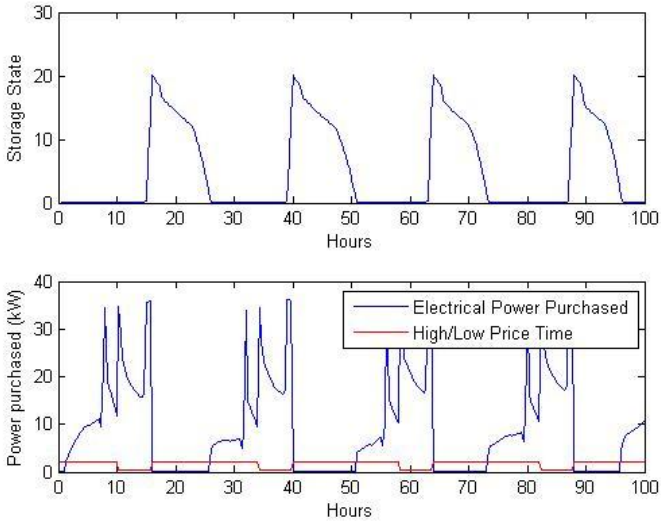

Figure 12: The battery state _ Battery storage

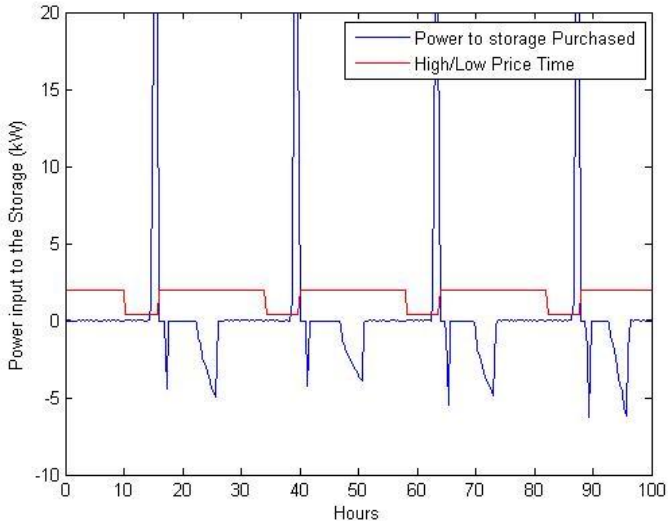

Figure 13: The power input _ Battery storage

2. Explain how the battery is utilized by the building and why.

In the figure 12 and the figure 13, we can see clearly that the battery is charged at the end of the low price period and is used during the high price period. This leads to the reduction of the total purchased electricity price.

3. Vary the storage capacity $\tilde{x}$ as well as the dissipation factor of the battery in a range you see appropriate, and explain the impact on the results. (You can compare the use of the battery as well as the total energy consumption)

In order to answer this question, we decided to simulate the system for different storage capacities (20, 50 and 100 kWh) and for different dissipation factors (0.9 and 0.98). Then, each time we calculate the total purchased energy in kWh and the total cost of the purchased energy in $.

	Storage capacity = 20		**Storage capacity = 50**		**Storage capacity = 100**	
Dissipation factor = 0.9	**2.7060e+03**	**296.2730**	**2.7060e+03**	**296.2730**	**2.7060e+03**	**296.2730**
Dissipation factor = 0.98	**2.7037e+03**	**283.6284**	**2.9167e+03**	**262.9018**	**3.3597e+03**	**245.4330**

From this table above, we can conclude that for a low dissipation (0.98), by increasing the storage capacity, we increase the total energy consumption but we reduce the total cost, because the system has a tendency to store more during the low price period. However, the increases in the total consumption can be explained by the dissipation.

On the other hand, for a high dissipation (0.9), the battery becomes unuseful and both the total purchased energy and the total cost of the purchased energy remain unchanged for different storage capacity.