Model Predictive Control ME-425
Prof. Colin Jones

Mini-Project
Building Climate Control

In this project you will implement an MPC controller for a building's heating system. You will design an MPC controller to minimize the total cost of operation that takes into account the prediction of the weather, and study the impact of additional heat storage.

## What is expected

At the end of the project you should hand in

- a working code: use the template for the function `main.m` and `shiftPred.m` which is provided. You might have to change the functions `simBuild.m, simBuildStorage.m` as well so provide a copy of it with your code. Make sure your code is working properly **WITHOUT MODIFICATION** before submitting it. If not, we won't debug it. We suggest you make sections in the main file for each question. You can also submit a separate file for each question.
- A small report with requested plots and answers to questions posed along the way

## Files

Please go to moodle, download and unpack the files `buildingProject.zip`. Make sure you have an updated version of YALMIP on your path. You will need a good solver for this project. We strongly recommend installing GUROBI by requesting an academic licence using your .epfl.ch email adress. Run the function `main.m` from the folder `buildingProject`; if it finishes successfully, you are ready to start.

## Getting acquainted - Building model

You are provided with a discrete-time state space model of a building which has the following form:

$$\begin{aligned} x_{k+1} &= Ax_k + B_u u_k + B_d d_k \\ y_k &= Cx_k \end{aligned}$$

The input of the model

$$u_k = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$$

is the electrical power input (kW) to the heating system of the three zones. The disturbance input

$$d_k = \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix}^T$$

constitutes the outside temperature ($^\circ$C), solar gains (kW), and internal gains (kW). The output of the model

$$y_k = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T$$

is the temperature in the three building zones. There are ten states in the model, however they do not have any physical interpretation. The sampling rate of the model is $T_s = 20$ min. The model parameters are provided in the MATLAB structure `ssM`. Moreover, the variable `refDist` (3x576) contain the values of the three disturbance inputs $d_k$ for a period of eight days, sampled at $T_s = 20$ min.

## Deliverables

1. plot the three disturbance inputs and explain anything noticeable *5 %*

## Constraint specification

The inputs and the outputs of the model are subject to constraints:

- $\forall k, u_{min} \leq u_k \leq u_{max}$,
- $\forall k, y_{min} \leq y_k \leq y_{max}$,

The input constraints capture the heating capacity of the HVAC system, while the output constraints ensure comfort in the building.

- $u_{max} = 15kW$
- $u_{min} = 0kW$
- $y_{max} = 26\,^{\circ}\mathrm{C}$
- $y_{min} = 22\,^{\circ}\mathrm{C}$

## First MPC controller

You are ready to design and test an output tracking MPC controller. Use YALMIP to create an optimizer object that solves the MPC problem at hand. Decide if you need to use terminal sets and terminal weights, and discuss your decision and the reason with the assistants before continuing.

Design any MPC controller that will regulate the outputs of the system from the given initial condition to, say

$$y_{ref} = \begin{bmatrix} 24 & 24 & 24 \end{bmatrix}^T.$$

This may be achieved by penalizing the difference between the output of the system, and the reference output. You may use a cost function of the form:

$$J = \sum_{k=1}^{N} (y_k - y_{ref})^T R (y_k - y_{ref})$$

where `N` is the prediction horizon of your MPC controller.

To simulate the system use the function `simBuild.m`. It takes the Yalmip optimizer instance as its second argument. Take a look at the documentation of the function. The syntax you end up using may look like this:

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

where `x0`, `d`, and `u` are sdpvar instances representing the initial state $x_0$, disturbance prediction over the horizon $d$, and the sequence of control inputs over the horizon $u$. $T$ is the simulation length in time-steps. As you can see, the disturbance will come as an input to the optimization problem (exactly like the initial condition). This means that the optimizer defined in the code above will solve the MPC problem at hand for a particular initial condition $x_0$ and a particular prediction of the disturbance $d$.

*Important Remark:* You are free to choose the exact form of the optimizer you use (cell or arrays, ordering of the inputs...) but be aware that the syntax has to be consistent between the functions `shiftPred.m`, `main.m` and `simBuild.m`. Therefore, the syntax in the simulation section of `shiftPred.m` (line 48 onwards) has to be adapted in accordance to the formulation of the optimizer in the function `main.m`. In the remainder of these instructions and in the code we propose using arrays to specify the inputs to the optimizer.

During the simulation, the controller will use updated predictions at each time-step for the weather. The MATLAB function `@shiftPred` outputs the predictions required by the optimizer over the horizon, at each time-step. Refer to the documentation of this function for further information. The last argument of the `simBuild.m` function is an option for the simulator. For now, just use 1, other options will be used later in the project.

*Warning:* You need to fix the same prediction horizon, N in the `shiftPred.m` function, as in the design of your MPC controller.

**Deliverables**

1. Explanation of the influence of the tuning parameters on the MPC scheme and choice of your tuning parameters *5 %*
2. Code generating the plots and comments on the plots *15 %*

## Economic MPC

Now you will design an Economic MPC controller for the building. The objective is to keep the temperature of the building zones within the comfort constraints, while minimizing our energy bill. Economic MPC uses an economic cost function, as opposed to the usual quadratic cost function used in regulation problems. Indeed the quadratic cost we used in the previous section does not actually reflect the true cost of the operation of the building. Therefore it can be preferable to use the "true" cost in the optimization. In this case, we will start by considering that we pay a fixed electricity price $c$ (\$/kWh). Modify the cost function to consider the true cost of operation. We take a price of $c = 0.2\$/kWh$.

*Remark:* By dropping the usual quadratic cost, guarantees of stability and recursive feasibility may be lost and other assumptions might be required. We will not discuss this here. Anyway, in the case of the building thermal regulation, regulation of the temperature is not exactly our goal. We rather just aim at maintaining acceptable indoor conditions under a minimum economic cost. The temperature is therefore entirely handled through constraints and the objective handles the cost of operation.

## Soft Constraints

To maintain the feasibility of the economic MPC optimization problem, you are advised to use soft constraints on the outputs (comfort constraints). Following the recommendation given in class, implement soft constraints on this problem.

To simulate your controller, you can again use the `simBuild.m` function with the last argument 1.

**Deliverables**

1. Choice of the cost function to penalize the slack variables and motivation for it *5 %*
2. Code generating the plots and comments on the plots *15 %*

## Variable Cost

In reality, the electricity prices are not constant, and most of the time have a periodic high price and low price periods. Economic MPC can be designed to account from this variation in the electricity prices and to shift the demand in order to minimize the total electricity consumption cost.

To implement variable prices, you are advised to add variable prices in your soft-constraint economic MPC by defining the `c` variable as a sdpvar instance over the prediction horizon. Make changes in the optimizer, such that the values of the variable cost are provided to the optimizer at each time instant. *Notice:* Normally, in your formulation, the price of electricity at each time-step will multiply the electricity consumption. This apparently yield a bilinear cost function. However, since the price of electricity will be given, it actually only is a linear cost function. Make sure you add a '+' sign before the name of the solver in the optimizer options (using sdpsettings) so the optimizer detects this.

```
ops = sdpsettings('verbose',1, 'solver', '+qpip');
```

To simulate your controller, you can again use the `simBuild.m` function with the last argument 2.

The variation in the electricity price is given below:

- $c_k = 0.2\$/kWh$ (between 00:00Hrs to 10:00Hrs, and 16:00Hrs to 24:00Hrs)
- $c_k = 0.04\$/kWh$ (between 10:00Hrs to 16:00Hrs)

You have to define the shifted prediction of the variable cost over the prediction horizon, at each time step, in the shiftPred function, similar to the shifted disturbance predictions already defined there.

*Warning:* Make sure that the second output of the shiftPred function is the shifted electricity cost (as already defined in the output argument list of the function).

**Deliverables**

1. plots of the response starting from the given initial condition and its explanation *10 %*

## Night Setbacks

One of the widely used techniques of reducing the energy consumption in commercial buildings is to use time-varying comfort constraints. This means that the controller is maintains the comfort constrained during office hours, whereas the constraints are relaxed outside office hours. Implement night-setbacks (time varying comfort constraints) on the outputs of your soft constraint variable cost economic MPC controller. This may be achieved by adding a new variable *sb* that you use to model the setback.

where `sb` is a sdpvar instance representing the relaxation in the output constraints over the prediction horizon, and is provided to the optimizer (i.e., it's not an optimization variable). This will also appear in the parameter list of the optimizer. The time-varying output constraints are given below:

- $y_{max} = 26 + 4 = 30\,^{\circ}\text{C}$ (between 00:00Hrs to 08:00Hrs, and 18:00Hrs to 24:00Hrs)
- $y_{max} = 26\,^{\circ}\text{C}$ (between 08:00Hrs to 18:00Hrs)
- $y_{min} = 22 - 4 = 18\,^{\circ}\text{C}$ (between 00:00Hrs to 08:00Hrs, and 18:00Hrs to 24:00Hrs)
- $y_{min} = 22\,^{\circ}\text{C}$ (between 08:00Hrs to 18:00Hrs)

You have to define the shifted time-varying off-set in the comfort constraint, over the prediction horizon at each time step, in the shiftPred function, similar to the shifted disturbance predictions already defined there.

*Warning:* Make sure that the third output of the shiftPred function is the shifted time-varying off-set of the comfort constraints (as already defined in the output argument list of the function).

To simulate your controller, you can again use the `simBuild.m` function, but with the last argument 3.

**Deliverables**

1. plots of the response starting from the given initial condition and explanation of the results *15 %*

## Battery Storage

In this part of the project you will study the effects of adding an electric storage device to the building. The battery serves as an energy buffer between the electricity grid and the building: when the power consumption of the HVAC system is denoted $u_k$ as previously and the power consumption from the grid is denoted $e_k$ then the difference between these will be stored in the battery: hence if $\sum_{i=1}^{3} u_k^i < e_k$ then the difference is used to charge the battery, conversely if $\sum_{i=1}^{3} u_k^i > e_k$ then the battery is discharged to supply the difference. This leads to the following model for the battery:

$$\tilde{x}_{k+1} = \alpha \tilde{x}_k + \beta v_k$$

where $e_k$ is the total electrical power consumption (in kW), $u_k$ is the power input to the heating system of the three zones, as previously defined, and $v_k = e_k - \sum_{i=1}^{3} u_k^i$ is the charging power of the battery. $\tilde{x}$ is the state of the battery storage, and represents the state-of-charge of the battery. The model parameters are provided in the MATLAB structure `ssModel`.

The inputs and states of the battery model have the following constraints:

- $\tilde{x}_{min} \leq \tilde{x} \leq \tilde{x}_{max}$
- $v_{min} \leq v \leq v_{max}$

The constraint parameter values are given below:

- $\tilde{x}_{max} = 20$ kWh
- $\tilde{x}_{min} = 0$ kWh
- $v_{max} = 20$ kW
- $v_{min} = -20$ kW

In addition, we assume we cannot reinject power into the grid, which means $e \geq 0$

*Note:* The maximum constraint on the value of the storage state $\tilde{x}_{max}$ represents the total storage capacity of the storage.

Add the storage and the corresponding constraints to the previously designed soft constrained economic MPC controller. Use the same variable cost and night-set back schedule.

*Note:* Now, the cost won't include the power inputs to the three building zones $u_k$, but only the total power input by the storage $e_k$.

You can simulate this controller using the `simBuildStorage.m` function. Refer to the documentation of the function for more details. The syntax you end up using may look this this. The storage will be considered empty at the beginning of the simulation. (`xb0=0`)

```
ops = sdpsettings('verbose',1, 'solver', '+gurobi');
controller = optimizer(constraints,objective,ops,[x0;xb0;d(:);cp(:);sb(:)],[u;v;e]);
[xt, yt, ut, t, et, xbt] = simBuildStorage( controller, T, @shiftPred, N);
```

**Deliverables**

1. plots of the response starting from the given initial condition *10 %*
2. Explain how the battery is utilized by the building and why. *5 %*
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) *15 %*

*Note:* The last question requires multiple simulation. If you experience too long simulation times, you can discuss this with the assistants. You could for example reduce the horizon (but in a reasonable range) or the sampling time of the problem. Mostly the performance of the solver will be paramount.