Delft University of Technology

SC42075 Modelling of Hybrid Systems

Assignment 2021

Group A16
Vivek Varma (5227828)
Justine Kroese (4547659)

June 18, 2021

Contents

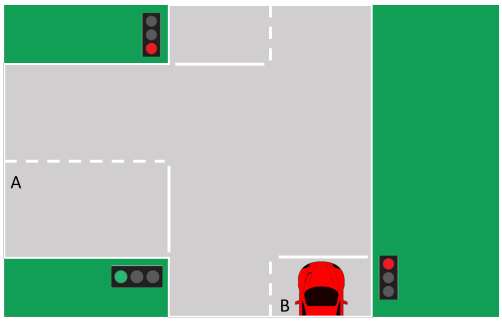
1 Part 1: Hybrid system example

In this chapter an intersection with traffic lights is represented as a hybrid automaton. First, the system itself is described and in after that this description is converted into a hybrid automaton.

1.1 Description of the system

This system that is going to be described is the intersection of the Hugo de Grootstraat and the Krakeelpolderweg in Delft, displayed in figure 1a. This is an intersection with traffic lights, where one light is used for all directions. So if the light is green the cars can choose whether to turn left or right or to drive straight ahead. A more schematic representation of the intersection can be found in figure 1b. The system will only take into account the car movements (pedestrians and cyclists are excluded).

(a) Photograph at intersection



(b) Schematic representation

Figure 1: System of an intersection with traffic lights

1.1.1 Variables and dynamics of the system

To model the intersection, it is split up into two roads. Road A is the road on the left and road B is the road straight ahead. The amount of cars on road A at a certain instance is modelled as y(t), the amount of road B as x(t). To make the system not too complicated, road A and road B will only have two states: closed and open. If the road is open, the light is green or orange, if the road is closed the light is red.

$y(t)$: the amount of cars waiting on road A
$x(t)$: the amount of cars waiting on the road B
$A(t)$: the state of road A $\in$ [Open,Closed]
$B(t)$: the state of road B $\in$ [Open,Closed]
$c(t)$: the time elapsed since both roads are closed

If the road is closed, the line of cars will grow. If the road is open the amount of cars in line will decrease. Only one road can be open at the same time. To switch which road is open, both roads will, for safety reasons, first be closed for 2 seconds. Because road A has two sides form which cars can arrive, the amount of cars is expected to grow faster.

This means there will be four states: The first state is where both roads are closed and $y(t) > 8$. The second state is where road A is open, road B is closed and $x \leq 5$. The third state is where both roads are closed and $x(t) > 5$. The last state is where road A is closed, road B is open and $y(t) \leq 8$.

1.2 The system as a hybrid automaton

Using the variables described is subsection 1.1.1, the hybrid automaton in figure 2 is constructed.

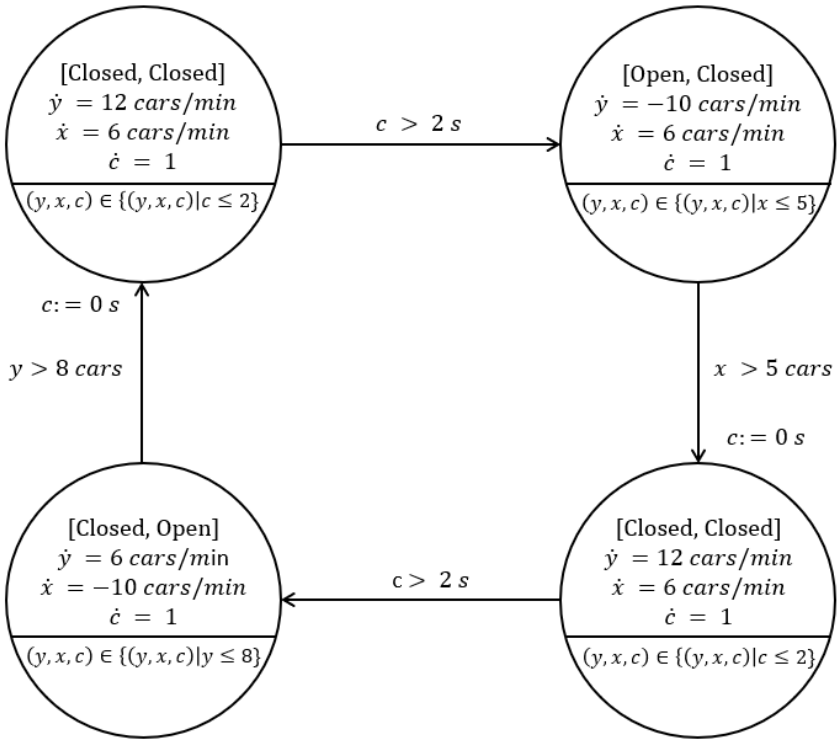


Figure 2: Hybrid automaton of intersection with traffic lights

The amount of cars in line and the amount of cars passing the open road are modelled as a continuous flow. This is of course a simplification of the reality, but it makes it the model easier to comprehend. When there are more than 8 cars waiting on road A or more than 5 on road B, that road needs to switch to the Open state. Before this can happen, the automaton first has to go through the state where both roads are closed, to make sure cars from the two roads will not hinder each other. After two seconds, the system gets into the next mode where the new road will open.

Because road A is considered the main road, the amount of cars necessary to open the road is set at a relatively lower amount (12 cars/min and more than 8 cars to open the road versus 6 cars/min and more than 5 cars).

2 Part 2: Energy management of microgrids

In this chapter, a microgrid which is connected to the main power grid in considered. A schematic representation of this system can be found in figure 3.

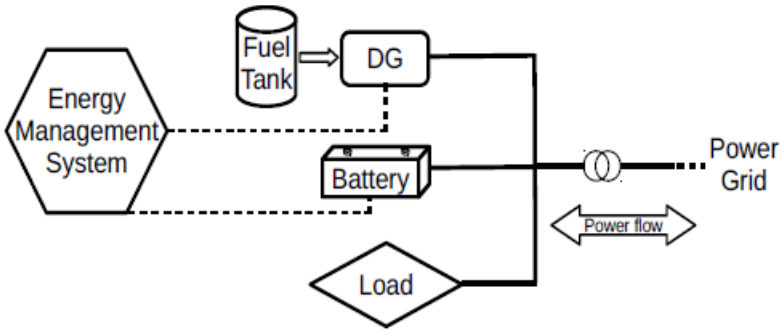


Figure 3: Schematic structure of the considered microgrid

The amount of power used by the load and the charging batteries is assumed equal to the power provided by the power grid, diesel generators, and the discharging batteries.

In the following sections, the grid will first be split up in the batteries and diesel generator. For each of those, a piecewise affine model will be created, which after that will be transformed into a mixed logical dynamical model. Thereafter, the models will be combined into one model for the whole system, which in the end will be solved as a model predictive control problem using the Gurobi optimiser.

2.1 Discrete-time Piecewise Affine (PWA) Model of a Battery

To determine a piecewise affine (PWA) model of a battery, it is important to first specify the parameters of the battery. These are stated in table 1.

Variables related to a battery	**Name**	**Range**	**Unit**
Stored energy	x_b	$[0,\bar{x_b}]$	kWh
Exchanged power	u_b	$[\underline{u_b},\bar{u_b}]$	kW
Operational mode (charge/discharge)	s_b	$\{0,1\}$	-
Charging efficiency	η_c	CONSTANT	-
Discharging efficiency	η_d	CONSTANT	-

Table 1: Parameters related to a battery

The state of the PWA model is the energy stored in the battery. The input is the exchanged power. The discharging of the battery is defined to have a positive sign and is connected to operational mode 0. Charging has a negative sign and is connected to operational mode 1. In one sampling instant T_s, the exchanged energy is is $T_s \cdot u_b$ kJ. This is than multiplied by the charging or discharging efficiency. This results in the PWA model:

$$x_b(k+1) = \begin{cases} x_b(k) - \eta_d T_s u_b(k) & u_b \geq 0 \quad (s_b(k) = 0) \\ x(b) - \eta_c T_s u_b(k) & u_b \leq 0 \quad (s_b(k) = 1) \end{cases}$$

2.2 Mixed Logical Dynamical (MLD) Model of a Battery

To determine the mixed logical dynamical (MLD) model of the battery, the following constraints are used:

$$0 \leq x_b \leq \bar{x_b}$$

$$\underline{u_b} \leq u_b \leq \bar{u_b}$$

$$if\ s_b(k) = 1\ u_b \leq 0$$
$$if\ s_b(k) = 0\ u_b > 0$$

To obtain the MLD model, we first take a look at the general model:

$$x(k+1) = Ax(k) + B_1u(k) + B_2d(k) + B_3z(k) + B_4$$

$$E_1x(k) + E_2u(k) + E_3\delta(k) + E_4z(k) \leq g_5$$

Looking at the model from subsection 2.1, it is clear that $x(k) = x_b(k)$ and $u(k) = u_b(k)$. $s_b(k)$ can be interpreted as the binary variable $\delta(k)$.

By using this $\delta(k)$, the PWA model can be rewritten as:

$$x(k+1) = x(k) - \eta_d T_s u(k) + (\eta_d - \eta_c) T_s \delta(k) u(k)$$

s.t.

$$u(k) \leq \bar{u_b}(1 - \delta(k))\ u(k) + \bar{u_b}\delta(k) \leq \bar{u_b}$$
$$u(k) \geq +(\underline{u_b} - \epsilon)\delta(k)\ \ - u(k) + (\underline{u_b} - \epsilon)\delta(k) \leq -\epsilon$$

The constraints can also be written in matrix form:

$$\begin{bmatrix} \bar{u_b} \\ \underline{u_b} - \epsilon \end{bmatrix} \delta(k) + \begin{bmatrix} 1 \\ -1 \end{bmatrix} u(k) \leq \begin{bmatrix} \bar{u_b} \\ -\epsilon \end{bmatrix}$$

In the constraints the ϵ is used to turn the strict inequality $u_b > 0$ into a non-strict equality.

In the equation above there is a product of two variables present: $\delta(k)u(k)$. This can be replaced by the new variable $z(k) = \delta(k)u(k)$. Then the formula for the state evolution is equal to:

$$x(k+1) = x(k) - \eta_d T_s u(k) + (\eta_d - \eta_c) T_s z(k)$$

The constraints are now equal to:

$$z(k) \leq \bar{u_b}\delta(k)$$
$$z(k) \geq \underline{u_b}\delta(k)$$

$$z(k) \leq u(k) - \underline{u_b}(1 - \delta(k))$$

$$z(k) \geq u(k) - \bar{u_b}(1 - \delta(k))$$

or in matrix form:

$$\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} z(k) + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} -\bar{u_b} \\ \underline{u_b} \\ -\underline{u_b} \\ \bar{u_b} \end{bmatrix} \delta(k) \leq \begin{bmatrix} 0 \\ 0 \\ -\underline{u_b} \\ \bar{u_b} \end{bmatrix}$$

The last constraint that has to written in vector form is $0 \leq x(k) \leq \bar{x_b}$:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} x(k) \leq \begin{bmatrix} \bar{x_b} \\ 0 \end{bmatrix}$$

By combining all these constraints the form of the general model can be obtained:

$$x(k+1) = \underbrace{1}_{A} \cdot x(k) + \underbrace{(-\eta_d T_s)}_{B_1} u(k) + \underbrace{0}_{B_2} \delta(k) + \underbrace{(\eta_d - \eta_c) T_s}_{B_3} z(k)$$

s.t.

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}}_{E_1} x(k) + \underbrace{\begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{E_2} u(k) + \underbrace{\begin{bmatrix} u_{\underline{b}} - \epsilon \\ \bar{u_b} \\ -\bar{u_b} \\ u_{\underline{b}} \\ -u_{\underline{b}} \\ \bar{u_b} \\ 0 \\ 0 \end{bmatrix}}_{E_3} \delta(k) + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}}_{E_4} z(k) \leq \underbrace{\begin{bmatrix} -\epsilon \\ \bar{u_b} \\ 0 \\ 0 \\ -u_{\underline{b}} \\ \bar{u_b} \\ \bar{x_b} \\ 0 \end{bmatrix}}_{g_5}$$

2.3 Modelling a Diesel Generator

The fuel consumption of the diesel generator is given by Equation 1.

$$f\left(u_{\mathrm{d}}(k)\right) = \begin{cases} u_{\mathrm{d}}^2(k) + 4 & \text{if } 0 \leq u_{\mathrm{d}}(k) < 2 \\ 4u_{\mathrm{d}}(k) & \text{if } 2 \leq u_{\mathrm{d}}(k) < 5 \\ -9.44u_{\mathrm{d}}^3(k) + 166.06u_{\mathrm{d}}^2(k) - 948.22u_{\mathrm{d}}(k) + 1790.28 & \text{if } 5 \leq u_{\mathrm{d}}(k) < 7 \\ -11.78u_{\mathrm{d}}(k) + 132.44 & \text{if } 7 \leq u_{\mathrm{d}}(k) < 9 \\ 4.01\left(u_{\mathrm{d}}(k) - 10.47\right)^2 + 17.79 & \text{if } 9 \leq u_{\mathrm{d}}(k) \leq 15, \end{cases} \tag{1}$$

where $f(u_d(k))$ is the consumed fuel of the diesel generator at time step k in [kg/h]. The value of $u_d(k)$ represents the output power of the diesel generator at time step k.
Now the task is to construct a Piecewise affine (PWA) model of the non-linear model in Equation 1. This PWA approximation ($\hat{f}\,[0, \bar{u}_d] \to \mathbb{R}$) consists of 4 regions where the fuel consumption curve of Equation 1 has to be approximated. This is defined as in Equation 2.

$$\hat{f}\left(u_{\mathrm{d}}(k)\right) = \begin{cases} a_1 + b_1 u_{\mathrm{d}}(k) & \text{if } 0 \leq u_{\mathrm{d}}(k) < u_1 \\ a_2 + b_2 u_{\mathrm{d}}(k) & \text{if } u_1 \leq u_{\mathrm{d}}(k) < u_2 \\ a_3 + b_3 u_{\mathrm{d}}(k) & \text{if } u_2 \leq u_{\mathrm{d}}(k) < u_3 \\ a_4 + b_4 u_{\mathrm{d}}(k) & \text{if } u_3 \leq u_{\mathrm{d}}(k) \leq 15, \end{cases} \tag{2}$$

where $u_1 = 5, u_2 = 6.5, u_3 = 11$.

To construct the best possible approximation, it needs to be ensured that the squared error through the intervals is minimised. Mathematically, this means that the cost function is:

$$c = \min \int_0^{\bar{u}_{\mathrm{d}}} \left(f\left(u_{\mathrm{d}}\right) - \hat{f}\left(u_{\mathrm{d}}\right)\right)^2 du_{\mathrm{d}} \tag{3}$$

A Genetic Algorithm (GA) has been used to find the minimum of Equation 3. We define Equation 2 and Equation 3 using the **piecewise**() function in MATLAB. After that, the integration function has been used to complete the cost function definition. The GA parameters defined explicitly are as follows:

GA parameter	Value
Population size	700
Maximum generations	1600
Elite count	25

Table 2: Parameters genetic algorithm

The results that were obtained (rounded to 4 decimal places) are equal to:

$$\{a_1, b_1\} = \{1.6318, 3.5392\},\ \{a_2, b_2\} = \{-85.6286, 20.8671\}$$

$$\{a_3, b_3\} = \{111.8701, -9.1653\},\ \{a_4, b_4\} = \{-207.2789, 19.6968\}$$

$$c = 128.0206$$

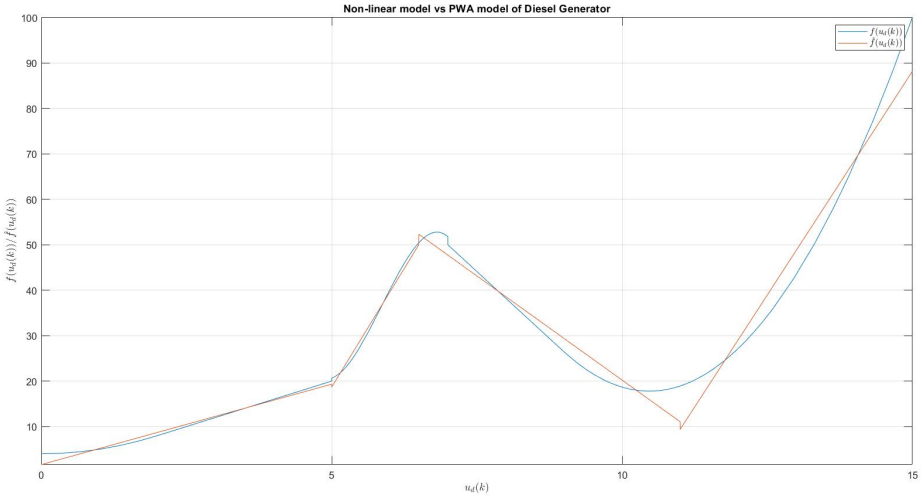


Figure 4: PWA Approximation of the non-linear Diesel Generator

Our PWA approximation is visualised in figure 4. Considering that there are only 4 PWA models available as an approximation, this approximation seems to be a good fit for the non-linear model.

2.4 Variable Limits of PWA Approximation

In this section there is the additional freedom that u_1, u_2, u_3 need not be fixed as they were previously. The cost function can now be re-modelled to have u_1, u_2, u_3 as variables to be optimised, in addition to a_i, b_i. This can be done in the piecewise function definition. The values of the previous section can be used as an initial guess, which means another optimisation algorithm can be used to find a better cost function value and thus better values for a_i, b_i, u_j.

The algorithm that is used now is Simulated Annealing, with the initial guess values for a_i, b_i *and* u_j from Question 2.3. Additionally, an upper and lower bound is defined, since u_j is bounded. Its lower bound is

0 and upper bound is 15.
Indeed with more freedom of optimisation, a better result in obtained. The results which were obtained (rounded to 4 decimal places) are equal to:

$$\{a_1, b_1\} = \{1.7111, 3.5224\},\ \{a_2, b_2\} = \{-85.6007, 20.7958\}$$

$$\{a_3, b_3\} = \{111.0235, -9.0577\},\ \{a_4, b_4\} = \{-207.4428, 19.705\}$$

$$u_1 = \{5.1091\},\ u_2 = \{6.8142\},\ u_3 = \{11.0738\},\ c = 126.5814$$

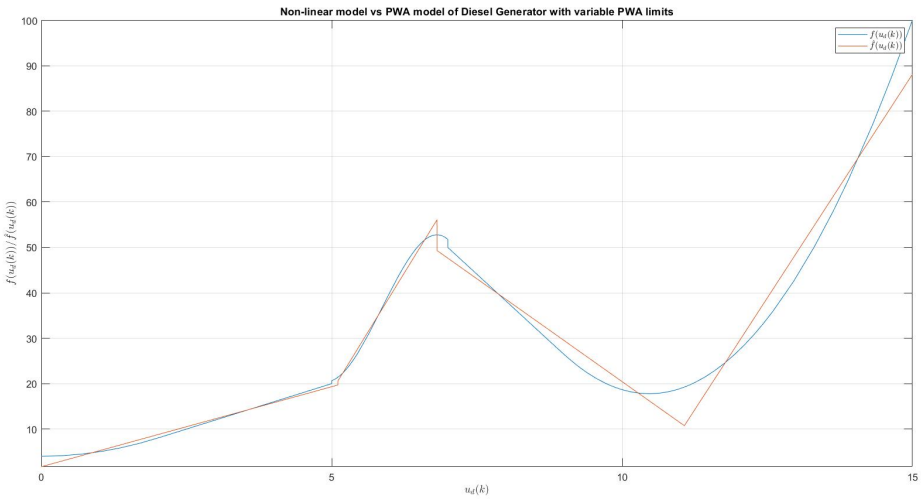


Figure 5: PWA Approximation of the Non-Linear Diesel Generator with Variable Limits

Our PWA approximation is visualised in figure 5. This does seem to be an even better fit. Minor differences can be noted in the PWA linear intercepts and slopes, but the major difference is the shifting of u_2 from 6.5 to 6.8. Thus with a greater degree of freedom, a lower cost function value is obtained. Thus this gives smaller errors and a better approximation of the non-linear diesel generator fuel consumption model.

2.5 Discrete-Time PWA Model of Diesel Generator

To determine a piecewise affine (PWA) model of the diesel generator, it is important to first specify the parameters of the generator. These are stated in table 3.

Variables related to a diesel generator	**Name**	**Range**	**Unit**
Remaining Fuel	x_d	$[\underline{x}_d, \bar{x}_d]$	kg
Generated power	u_d	$[0, \bar{u}_d]$	kW
Operational mode (on/off)	s_d	$\{0,1\}$	-
Filling rate of fuel tank	R_f	CONSTANT	-

Table 3: Parameters related to a diesel generator

The state of the PWA model is the remaining fuel in the diesel generator. The input is the generated power. The use of the generator is defined consumes fuel and is connected to operational mode 0. Refilling is constantly adding fuel to the system, and the absence of consumed fuel to generate power is connected

to operational mode 1. In one step(one hour), the fuel tank is constantly refilled by R_f irrespective of the switch state. This results in the PWA model:

$$x_d(k+1) = \begin{cases} x_d(k) + R_f T_s & \text{if } s_d(k) = 0 \quad (u_d(k) = 0) \\ x_d(k) + R_f T_s - (a_1 + b_1 \left(u_{\mathrm{d}}(k)\right) T_s & \text{if } s_d(k) = 1 \quad 0 \leq u_{\mathrm{d}}(k) < u_1 \\ x_d(k) + R_f T_s - (a_2 + b_2 \left(u_{\mathrm{d}}(k)\right) T_s & \text{if } s_d(k) = 1 \quad u_1 \leq u_{\mathrm{d}}(k) < u_2 \\ x_d(k) + R_f T_s - (a_3 + b_3 \left(u_{\mathrm{d}}(k)\right) T_s & \text{if } s_d(k) = 1 \quad u_2 \leq u_{\mathrm{d}}(k) < u_3 \\ x_d(k) + R_f T_s - (a_4 + b_4 \left(u_{\mathrm{d}}(k)\right) T_s & \text{if } s_d(k) = 1 \quad u_3 \leq u_{\mathrm{d}}(k) < 15 \end{cases} \tag{4}$$

2.6 Mixed Logical Dynamical (MLD) Model of a Diesel Generator

To determine the mixed logical dynamical (MLD) model of the battery, the following constraints are used:

$$\underline{x_d} \leq x_d \leq \bar{x_d}$$

$$0 \leq u_b \leq \bar{u_b}$$

$$if\ s_d(k) = 1\ u_d > 0$$

$$if\ s_d(k) = 0\ u_d = 0$$

To obtain the MLD model, we first take a look at the general model:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 d(k) + B_3 z(k) + B_4$$

$$E_1 x(k) + E_2 u(k) + E_3 \delta(k) + E_4 z(k) \leq g_5$$

Looking at the model from subsection 2.5, it is clear that $x(k) = x_d(k)$ and $u(k) = u_d(k)$. $s_d(k)$ can be interpreted as the binary variable $\delta(k)$. To account for the four different operational modes in the PWA model, four δ's are introduced. Only only of these can be nonzero at the same time. If all are zero, the generator is not generating any power.

$$\begin{array}{ll} \delta_1(k) = 1 & 0 < u(k) < u_1 \\ \delta_2(k) = 1 & u_1 \leq u(k) < u_2 \\ \delta_3(k) = 1 & u_2 \leq u(k) < u_3 \\ \delta_4(k) = 1 & u_3 \leq u(k) < 15 \end{array} \tag{5}$$

$$\text{s.t. } \delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) \leq 1$$

By using these $\delta(k)$, the PWA model can be rewritten as:

$$x(k+1) = x(k) + R_f T_s - (a_1 + b_1 u(k)) T_s \delta_1(k) - (a_2 + b_2 u(k)) T_s \delta_2(k) - (a_3 + b_3 u(k)) T_s \delta_3(k) - (a_4 + b_4 u(k)) T_s \delta_4(k)$$

s.t.

$$0 \leq u(k) \leq \bar{u}_d + \delta_1(k)(u_1 - \bar{u}_d - \epsilon)$$

$$\delta_2(k) u_1 \leq u(k) \leq \bar{u}_d + \delta_2(k)(u_2 - \bar{u}_d - \epsilon)$$

$$\delta_3(k) u_2 \leq u(k) \leq \bar{u}_d + \delta_3(k)(u_3 - \bar{u}_d - \epsilon)$$

$$\delta_4(k) u_3 \leq u(k) \leq 15$$

$$u(k) \leq (\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k)) \bar{u}_d$$

$$\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) \leq 1$$

$$\underline{x_d} \leq x_d \leq \bar{x_d}$$

The equation with its constraints can also be written in matrix-vector form. Then the model is equal to:

$$x(k+1) = x(k) + R_f T_s - T_s \mathbf{a} \boldsymbol{\delta_d}(k) - T_s \mathbf{b} \boldsymbol{\delta_d}(k) u(k)$$

s.t

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} x(k) + \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -u_1 + \bar{u}_d + \epsilon & 0 & 0 & 0 \\ 0 & u_1 & 0 & 0 \\ 0 & -u_2 + \bar{u}_d + \epsilon & 0 & 0 \\ 0 & 0 & u_2 & 0 \\ 0 & 0 & -u_3 + \bar{u}_d + \epsilon & 0 \\ 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 \\ -\bar{u}_d & -\bar{u}_d & -\bar{u}_d & -\bar{u}_d \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \delta_d(k) \leq \begin{bmatrix} 0 \\ \bar{u}_d \\ 0 \\ \bar{u}_d \\ 0 \\ \bar{u}_d \\ 0 \\ 15 \\ 0 \\ 1 \\ \bar{x}_d \\ -\underline{x_d} \end{bmatrix}$$

where $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix}$, $\boldsymbol{\delta_d} = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix}$

Because the left sides of the constraints and the right side of the last u(k) constraint are already non-strict equalities, there is no ϵ needed there.

In the equation above there is a product of two variables present: $\delta_d(k)u(k)$. This can be replaced by the new variable

$$\mathbf{z_d}(k) = \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \end{bmatrix} = \boldsymbol{\delta_d}(k) u(k) = \begin{bmatrix} \delta_1(k)u(k) \\ \delta_2(k)u(k) \\ \delta_3(k)u(k) \\ \delta_4(k)u(k) \end{bmatrix}$$

.

Then the formula for the state evolution is equal to:

$$x(k+1) = x(k) + R_f T_s - T_s \mathbf{a} \boldsymbol{\delta_d}(k) - T_s \mathbf{b} \mathbf{z_d}(k)$$

The constraints are now equal to:

$$z_i(k) \leq u_i \delta_i(k)$$

$$z_i(k) \geq u_{i-1} \delta_i(k)$$

$$z_i(k) \leq u(k) - \underline{u_d}(1 - \delta_i(k))$$

$$z_i(k) \geq u(k) - \bar{u_d}(1 - \delta_i(k))$$

where $i \in [1, 4]$, $u_0 = 0$ and $u_4 = 15$.

Here u_i and u_{i-1} are the limits of $u(k)$ in the different operational modes. Because $\underline{u_d} = 0$, the constraints can be simplified to:

$$z_i(k) \leq u_i \delta_i(k)$$

$$z_i(k) \geq u_{i-1} \delta_i(k)$$

$$z_i(k) \leq u(k)$$

$$z_i(k) \geq u(k) - \bar{u_d}(1 - \delta_i(k))$$

or in matrix form:

$$\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} z_i(k) + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} -u_i \\ u_{i-1} \\ 0 \\ \bar{u_d} \end{bmatrix} \delta_i(k) \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \bar{u_d} \end{bmatrix}$$

The last constraint that has to be written in vector form is $\underline{x_d} \leq x(k) \leq \bar{x_d}$:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} x(k) \leq \begin{bmatrix} \bar{x_d} \\ \underline{x_d} \end{bmatrix}$$

By combining all these constraints the form of the general model can be obtained:

$$x(k+1) = \underbrace{1}_{A} \cdot x(k) + \underbrace{0}_{B_1} u(k) + \underbrace{(-T_s \mathbf{a})}_{B_2} \delta_d(k) + \underbrace{(-T_s \mathbf{b})}_{B_3} z_d(k) + \underbrace{R_f T_s}_{B_4}$$

s.t.

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}}_{E_1} x(k) + \underbrace{\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{E_2} u(k) + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ -u_1 + \bar{u}_d + \epsilon & 0 & 0 & 0 \\ 0 & u_1 & 0 & 0 \\ 0 & -u_2 + \bar{u}_d + \epsilon & 0 & 0 \\ 0 & 0 & u_2 & 0 \\ 0 & 0 & -u_3 + \bar{u}_d + \epsilon & 0 \\ 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 \\ -\bar{u_d} & -\bar{u_d} & \bar{u_d} & \bar{u_d} \\ 1 & 1 & 1 & 1 \\ -u_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \bar{u_d} & 0 & 0 & 0 \\ 0 & -u_2 & 0 & 0 \\ 0 & u_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \bar{u_d} & 0 & 0 \\ 0 & 0 & -u_3 & 0 \\ 0 & 0 & u_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{u_d} & 0 \\ 0 & 0 & 0 & -15 \\ 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bar{u_d} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{E_3} \delta(k)$$

$$+\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{E_4} z(k) \leq \underbrace{\begin{bmatrix} 0 \\ \bar{u}_d \\ 0 \\ \bar{u}_d \\ 0 \\ \bar{u}_d \\ 0 \\ 15 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \bar{u_d} \\ 0 \\ 0 \\ 0 \\ \bar{u_d} \\ 0 \\ 0 \\ 0 \\ \bar{u_d} \\ 0 \\ 0 \\ 0 \\ \bar{u_d} \\ \bar{x_d} \\ \bar{x_d} \end{bmatrix}}_{g_5}$$

2.7 System model

For the system as a whole, two batteries and one diesel generator are considered. Their MLD models are repeated below (including corresponding subscripts).

$$x_{b_1}(k+1) = A_{b_1}x_{b_1}(k) + B_{b_1,1}u_{b_1}(k) + B_{b_1,2}\delta_{b_1}(k) + B_{b_1,3}z_{b_1}(k) + B_{b_1,4}$$
$$x_{b_2}(k+1) = A_{b_2}x_{b_2}(k) + B_{b_2,1}u_{b_2}(k) + B_{b_2,2}\delta_{b_2}(k) + B_{b_2,3}z_{b_2}(k) + B_{b_2,4}$$
$$x_d(k+1) = A_d x_d(k) + B_{d,1}u_d(k) + B_{d,2}\delta_d(k) + B_{d,3}z_d(k) + B_{d,4}$$

where $B_{b_1,4}$ and $B_{b_2,4}$ are zero subject to the constraints:

$$E_{b_1,1}x_{b_1}(k) + E_{b_1,2}u_{b_1}(k) + E_{b_1,3}\delta_{b_1}(k) + E_{b_1,4}z_{b_1}(k) \leq g_{b_1,5}$$
$$E_{b_2,1}x_{b_2}(k) + E_{b_2,2}u_{b_2}(k) + E_{b_2,3}\delta_{b_2}(k) + E_{b_2,4}z_{b_2}(k) \leq g_{b_2,5}$$
$$E_{d,1}x_d(k) + E_{d,2}u_d(k) + E_{d,3}\delta_d(k) + E_{d,4}z_d(k) \leq g_{d,5}$$

To make a MLD model of the whole system, the matrices can be stacked together in (block diagonal) matrices. The model we get then is as follows:

$$x(k+1) = \underbrace{\begin{bmatrix} A_{b_1} & 0 & 0 \\ 0 & A_{b_2} & 0 \\ 0 & 0 & A_d \end{bmatrix}}_{A} \begin{bmatrix} x_{b_1}(k) \\ x_{b_2}(k) \\ x_d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{b_1,1} & 0 & 0 \\ 0 & B_{b_2,1} & 0 \\ 0 & 0 & B_{d,1} \end{bmatrix}}_{B_1} \begin{bmatrix} u_{b_1}(k) \\ u_{b_2}(k) \\ u_d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{b_1,2} & 0 & 0 \\ 0 & B_{b_2,2} & 0 \\ 0 & 0 & B_{d,2} \end{bmatrix}}_{B_2} \begin{bmatrix} \delta_{b_1}(k) \\ \delta_{b_2}(k) \\ \delta_d(k) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} B_{b_1,3} & 0 & 0 \\ 0 & B_{b_2,3} & 0 \\ 0 & 0 & B_{d,3} \end{bmatrix}}_{B_3} \begin{bmatrix} z_{b_1}(k) \\ z_{b_2}(k) \\ z_d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{b_1,4}(k) \\ B_{b_2,4}(k) \\ B_{d,4}(k) \end{bmatrix}}_{B_4}$$

s.t.

$$\underbrace{\begin{bmatrix} E_{b_1,1} & 0 & 0 \\ 0 & E_{b_2,1} & 0 \\ 0 & 0 & E_{d,1} \end{bmatrix}}_{E_1} \begin{bmatrix} x_{b_1}(k) \\ x_{b_2}(k) \\ x_d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} E_{b_1,2} & 0 & 0 \\ 0 & E_{b_2,2} & 0 \\ 0 & 0 & E_{d,2} \end{bmatrix}}_{E_2} \begin{bmatrix} u_{b_1}(k) \\ u_{b_2}(k) \\ u_d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} E_{b_1,3} & 0 & 0 \\ 0 & E_{b_2,3} & 0 \\ 0 & 0 & E_{d,3} \end{bmatrix}}_{E_3} \begin{bmatrix} \delta_{b_1}(k) \\ \delta_{b_2}(k) \\ \delta_d(k) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} E_{b_1,4} & 0 & 0 \\ 0 & E_{b_2,4} & 0 \\ 0 & 0 & E_{d,4} \end{bmatrix}}_{E_4} \begin{bmatrix} z_{b_1}(k) \\ z_{b_2}(k) \\ z_d(k) \end{bmatrix} \leq \underbrace{\begin{bmatrix} g_{b_1,5} \\ g_{b_2,5} \\ g_{d,5} \end{bmatrix}}_{g_5}$$

This MLD model has also been implemented in MATLAB. This implementation can be found in Appendix C.

2.8 Defining the cost function

The cost function is described as given below:

$$J(k) = \sum_{j=0}^{N_p-1} (\sum_{i=1}^{N_b} W_{b,i}|\Delta s_{b,i}(k+j)| + W_d|\Delta s_d(k+j)|) - W_{fuel}(x_d(k+N_p) - x_d(k))$$

$$-W_e \sum_{i=1}^{N_b} (x_{b,i}(k+N_p) - x_{b,i}(k)) + \sum_{j=0}^{N_p-1} P_{imp}(k+j)C_e(k+j)$$

where $P_{imp}(k)$ is the imported power to the microgrid at time step k and $C_e(k)$ is the price (benefit) of importing electricity to (exporting electricity from) the microgrid at time step k.

To determine the optimal Model predictive control (MPC) input sequence for a given sample step k, the MLD needs to be transformed into a mixed-integer linear programming problem (MILP). For this we need the values of $u(k)$, $\delta(k)$ and $z(k)$ up until $k + N_p - 1$, where N_p is the prediction horizon. However, we only have the values up until $k + N_c - 1$, where N_c is the control horizon. That is why the values from

$k + N_p$ until $k + N_c - 1$ are kept equal to the value at $k + N_p - 1$. In matrix form it looks like this:

$$\hat{u}(k) = \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_c-1) \\ u(k+N_c) \\ \vdots \\ u(k+N_p-1) \end{bmatrix} = \begin{bmatrix} I_{N_u} & 0 & \cdots & 0 \\ 0 & I_{N_u} & \vdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & I_{N_u} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & I_{N_u} \end{bmatrix} \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_c-1) \end{bmatrix} = K_u \tilde{u}(k)$$

where N_u is the length of $u(k)$. The same can be done for $\hat{\delta}(k) = K_\delta \tilde{\delta}(k)$ and $\hat{z}(k) = K_z \tilde{z}(k)$

The three vectors $\tilde{u}(k)$, $\tilde{\delta}(k)$ and $\tilde{z}(k)$ can also be combined in one matrix as:

$$\tilde{V}(k) = \begin{bmatrix} \tilde{u}(k) & \tilde{\delta}(k) & \tilde{z}(k) \end{bmatrix}$$

To be able to solve the problem, we have to rewrite both the MLD model and the cost function.

2.8.1 Rewriting the MLD model

The MLD model can be rewritten by using successive substitution:

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) + B_4 \\ x(k+2) &= A(Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) + B_4) + B_1 u(k+1) + B_2 \delta(k+1) + B_3 z(k+1) + B_4 \\ &\vdots \\ x(k+j) &= A^j x(k) + \sum_{i=0}^{j-1} A^{j-i-1}(B_1 u(k+i) + B_2 \delta(k+i) + B_3 z(k+i) + B_4) \end{aligned}$$

This can also be written in matrix format:

$$\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+N_p) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}}_{M_2} x(k) + \underbrace{\begin{bmatrix} B_1 & 0 & \cdots & 0 \\ AB_1 & B_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_1 & A^{N_p-2}B_1 & \cdots & B_1 \end{bmatrix}}_{T_1} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix} +$$

$$\underbrace{\begin{bmatrix} B_2 & 0 & \cdots & 0 \\ AB_2 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_2 & A^{N_p-2}B_2 & \cdots & B_2 \end{bmatrix}}_{T_2} \begin{bmatrix} \delta(k) \\ \delta(k+1) \\ \vdots \\ \delta(k+N_p-1) \end{bmatrix} + \underbrace{\begin{bmatrix} B_3 & 0 & \cdots & 0 \\ AB_3 & B_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B_3 & A^{N_p-2}B_3 & \cdots & B_3 \end{bmatrix}}_{T_3} \begin{bmatrix} z(k) \\ z(k+1) \\ \vdots \\ z(k+N_p-1) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} B_4 \\ AB_4 + B_4 \\ \vdots \\ A^{N_p-1}B_4 + A^{N_p-2}B_4 + \cdots + B_4 \end{bmatrix}}_{M_3}$$

So more compactly written this is equal to:

$$\hat{x}(k) = M_2 x(k) + T_1\hat{u}(k) + T_2\hat{\delta}(k) + T_3\hat{z}(k) + M_3 = M_2 x(k) + T_1 K_u \tilde{u}(k) + T_2 K_\delta \tilde{\delta}(k) + T_3 K_z \tilde{z}(k) + M_3$$

Now by using the $\tilde{V}(k)$ that was previously described:

$$\hat{x}(k) = M_2 x(k) + \begin{bmatrix} T_1 K_u & T_2 K_\delta & T_3 K_z \end{bmatrix} \tilde{V}(k) + M_3 = M_2 x(k) + M_1 \tilde{V}(k) + M_3$$

We need to follow a similar procedure as shown above to generalize the constraint equations and group together the variables in terms of $\tilde{V}(k)$. The constraints are vital and simultaenous to the above derivation since in each constraint step we get the values of $\hat{\delta}$ and $\hat{z}$. The constraint equations follow the pattern:

$$E_1 x(k) + E_2 u(k) + E_3\hat{\delta}(k) + E_4\hat{z}(k) \leq g_5$$

$$E_1 A\hat{x}(k) + E_1 B_1 u(k) + E_1 B_2\hat{\delta}(k) + E_1 B_3\hat{z}(k) + E_2 u(k+1) + E_3\hat{\delta}(k+1) + E_4\hat{z}(k+1) \leq g_5 - E_1 B_4$$

$$\vdots$$

$$\begin{aligned} &E_1 A^{j-1} x(k) + E_1 A^{j-2} B_1 u(k) + E_1 A^{j-3} B_1 u(k+1) + \cdots + E_1 B_1 u(k+j-2) + E_2 u(k+j-1) + \\ &E_1 A^{j-2} B_2\hat{\delta}(k) + E_1 A^{j-3} B_2\hat{\delta}(k+1) + \cdots + E_1 B_2\hat{\delta}(k+j-2) + E_3\hat{\delta}(k+j-1) + \\ &E_1 A^{j-2} B_3\hat{z}(k) + E_1 A^{j-3} B_3\hat{z}(k+1) + \cdots + E_1 B_3\hat{z}(k+j-2) + E_4\hat{z}(k+j-1) \\ &\leq g_5 - E_1 B_4 - E_1 A B_4 - \cdots - E_1 A^{j-2} B_4 \end{aligned}$$

This could also be written in a generalised form as shown below,

$$\underbrace{\begin{bmatrix} E_1 & 0 & \dots & 0 & 0 \\ 0 & E_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & E_1 & 0 \\ 0 & 0 & \dots & 0 & I \\ 0 & 0 & \dots & 0 & -I \end{bmatrix}}_{\hat{E}_1} \begin{bmatrix} x(k) \\ x(k+1) \\ \vdots \\ x\left(k+N_p-1\right) \\ x\left(k+N_p\right) \end{bmatrix} + \underbrace{\begin{bmatrix} E_2 & 0 & \dots & 0 \\ 0 & E_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_2 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_2} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u\left(k+N_p-1\right) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} E_3 & 0 & \dots & 0 \\ 0 & E_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_3 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_3} \begin{bmatrix} \delta(k) \\ \delta(k+1) \\ \vdots \\ \delta\left(k+N_p-1\right) \end{bmatrix} + \underbrace{\begin{bmatrix} E_4 & 0 & \dots & 0 \\ 0 & E_4 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_4 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_4} \begin{bmatrix} z(k) \\ z(k+1) \\ \vdots \\ z\left(k+N_p-1\right) \end{bmatrix} \leq \underbrace{\begin{bmatrix} g_5 \\ g_5 \\ \vdots \\ g_5 \\ \bar{x} \\ -\underline{x} \end{bmatrix}}_{\hat{g}_5}$$

$$\hat{E}_1\hat{x}(k-1) + \hat{E}_2\hat{u}(k) + \hat{E}_3\hat{\delta}(k) + \hat{E}_4\hat{z}(k) \leq \hat{g}_5$$

Additionally, we keep in mind that the states are bounded, and that the final state(x_{eq}) must be within the limits of $\underline{x} \leq x_{eq} \leq \bar{x}$. Correspondingly, additional rows are added to the matrices to fit this constraint equation. Although we have written this equation in this manner, we need to find a relation which incorporates the matrix $\tilde{V}_k$ as well as the relation between consecutive states.Now $\hat{x}(k-1)$ is written in terms of $\tilde{V}(k)$(according to the previous definition) and $x(k)$ as follows:

$$\hat{x}(k-1) = \begin{bmatrix} 0 \\ M_1 \end{bmatrix} \tilde{V}(k) + \begin{bmatrix} I \\ M_2 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ M_3 \end{bmatrix}$$

Now, we need to rewrite it in the form $F_1\tilde{V}(k) \leq F_2 + F_3 x(k)$

$$\underbrace{(\hat{E}_1 \begin{bmatrix} 0 \\ M_1 \end{bmatrix} + [\ \hat{E}_2 K_u \quad \hat{E}_3 K_\delta \quad \hat{E}_4 K_z\ ])}_{F_1} \tilde{V}(k) \leq \underbrace{\hat{g}_5 - \hat{E}_1 \begin{bmatrix} 0 \\ M_3 \end{bmatrix}}_{F_2} \underbrace{-\hat{E}_1 \begin{bmatrix} I \\ M_2 \end{bmatrix}}_{F_3} x(k)$$

2.8.2 Rewriting the Cost Function

We have the cost function

$$J(k) = \sum_{j=0}^{N_p-1} (\sum_{i=1}^{N_b} W_{b,i}|\Delta s_{b,i}(k+j)| + W_d|\Delta s_d(k+j)|) - W_{fuel}(x_d(k+N_p) - x_d(k))$$

$$-W_e \sum_{i=1}^{N_b}(x_{b,i}(k+N_p) - x_{b,i}(k)) + \sum_{j=0}^{N_p-1} P_{imp}(k+j)C_e(k+j)$$

where

$$P_{\text{imp}}\,(k+j) = P_{\text{load}}\,(k+j) - u_{\text{d}}(k+j) - \sum_{i=1}^{N_{\text{b}}} u_{\text{b},i}(k+j), \quad \forall j.$$

We need to rewrite this into a MILP form which can be used for optimisation of our MPC problem. Before we substitute the matrices M and F calculated previously, we first need to get our cost function into a linear form. We will do this term by term. The question 2.7 mentions creation of a model for 1 generator and 2 batteries. This code and the following operations will be performed with $N_b = 2$. The first part we address is $\sum_{j=0}^{N_p-1} P_{imp}(k+j)C_e(k+j)$.

$$\sum_{j=0}^{N_p-1} P_{imp}(k+j)C_e(k+j) = \sum_{j=0}^{N_p-1} (P_{\text{load}}\,(k+j) - u_{\text{d}}(k+j) - \sum_{i=1}^{N_{\text{b}}} u_{\text{b},i}(k+j))C_e(k+j)$$

where both $\tilde{P}_{load}$ and $\tilde{C}_e$ values are known. Therefore, their product can be written in a matrix form- $\tilde{C}_e^T \tilde{P}_{load}$. $-u_{\text{d}}(k+j) - \sum_{i=1}^{N_{\text{b}}} u_{\text{b},i}(k+j)$ can be written as $[-1\ -1\ -1]u(k+j)$ where $u(k+j) = [u_{b,1}(k+j)\ u_{b,2}(k+j)\ u_d(k+j)]^T$. Therefore,

$$\sum_{j=0}^{N_p-1} (P_{\text{load}}\,(k+j) - u_{\text{d}}(k+j) - \sum_{i=1}^{N_{\text{b}}} u_{\text{b},i}(k+j))C_e(k+j)$$

$$= \tilde{C}_e(k)^T \tilde{P}_{load}(k) + \underbrace{\tilde{C}_e(k)^T \begin{bmatrix} -1 & -1 & -1 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -1 & -1 & -1 \end{bmatrix}}_{C_u} \hat{u}(k)$$

where $\hat{u}(k) = K_u \tilde{u}(k)$ as previously defined. The next terms we focus on are

$$-W_{fuel}(x_d(k+N_p) - x_d(k)) - W_e \sum_{i=1}^{N_b}(x_{b,i}(k+N_p) - x_{b,i}(k))$$

$$= -[W_e\ W_e\ W_{fuel}]x(k) + \underbrace{[0\ \cdots\ 0\ W_e\ W_e\ W_{fuel}]}_{C_x} \hat{x}(k)$$

where $x(k) = [x_{b,1}(k)\ x_{b,2}(k)\ x_d(k)]^T$ and $\hat{x}(k) = [x(k+1)\ x(k+2)\ \cdots\ x(k+N_p)]^T$ Now we tackle the final 2 terms- the 1 norms.

$$\sum_{j=0}^{N_p-1}(\sum_{i=1}^{N_b} W_{b,i}|\Delta s_{b,i}(k+j)| + W_d|\Delta s_d(k+j)|)$$

$$= \sum_{j=0}^{N_p-1} W_{b,1}|\Delta s_{b,1}(k+j)| + W_{b,2}|\Delta s_{b,2}(k+j)| + W_d|\Delta s_d(k+j)|$$

The trick to linearize these terms and get it into a linear form is using slack variables. Using the previous definitions of δ, we take $s_d(k) = \delta_{d1} + \delta_{d2} + \delta_{d3} + \delta_{d4}$.

$$\Delta s(k) = \begin{bmatrix}\Delta s_1(k)\\ \Delta s_2(k)\\ \Delta s_3(k)\end{bmatrix} = \begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&1&1&1\end{bmatrix}\delta(k) + \begin{bmatrix}-1&0&0&0&0&0\\0&-1&0&0&0&0\\0&0&-1&-1&-1&-1\end{bmatrix}\delta(k-1)$$

Using the definition of the 1 norm we can also write

$$= \sum_{j=0}^{N_p-1} W_{b,1}|\Delta s_{b,1}(k+j)| + W_{b,2}|\Delta s_{b,2}(k+j)| + W_d|\Delta s_d(k+j)|$$

as

$$\left\| \underbrace{\begin{bmatrix}W_{b,1}&0&0&0&0&0\\0&W_{b,2}&0&0&0&0\\0&0&W_d&W_d&W_d&W_d\end{bmatrix}}_{W_\delta}\delta(k+j) + \begin{bmatrix}-W_{b,1}&0&0&0&0&0\\0&-W_{b,2}&0&0&0&0\\0&0&-W_d&-W_d&-W_d&-W_d\end{bmatrix}\delta(k+j-1)\right\|_1$$

=

$$\left\| \underbrace{\begin{bmatrix}W_\delta&0&\cdots&\cdots&0\\-W_\delta&W_\delta&0&\cdots&0\\0&-W_\delta&W_\delta&\cdots&0\\\vdots&\vdots&\ddots&\ddots&\vdots\\0&\cdots&\cdots&-W_\delta&W_\delta\end{bmatrix}}_{C_{\delta 1}}\hat{\delta}(k) + \underbrace{\begin{bmatrix}-W_\delta\\0\\\vdots\\0\end{bmatrix}}_{C_{\delta 2}}\delta(k-1)\right\|_1$$

where $\hat{\delta}(k) = [\delta(k)\ \cdots\ \delta(k+N_p-1)]^T$. We can now take $C_{\delta 1}\hat{\delta}(k) + C_{\delta 2}\delta(k-1) := \theta$ and our optimisation problem can be written as

$$\min_{\rho,\theta\epsilon\mathbb{R}^n}[1\ \cdots 1]p(k)$$

subject to

$$-p \leq C_{\delta 1}\hat{\delta}(k) + C_{\delta 2}\delta(k-1) \leq p$$

in addition to the standard constraints on $C_{\delta 1}\hat{\delta}(k) + C_{\delta 2}\delta(k-1)$. We can thus re-write our cost function as

$$J(k) = [1\ \cdots 1]p(k) + C_x\hat{x}(k) + C_u\hat{u}(k) + [W_e\ W_e\ W_{fuel}]x(k) + \tilde{C}_e(k)^T P_{load}(k)$$

2.8.3 Optimising the Modified Cost Function using the Modified MLD expressions

Now that we have a linear cost function, we can substitute the expressions for $\hat{x}(k) = M_1 x(k) + M_2 \tilde{V}(k) + M_3$ and $\hat{u}(k) = K_u \tilde{u}(k)$ which have been previously derived. Keep in mind that the objective function has to find optimal values of $\tilde{V}$ and $\tilde{\rho}$.

$$\min\ J(k) = \min\ [1\ \cdots 1]p(k) + C_x M_1 \tilde{V}(k) + C_x M_2 x(k) + C_x M_3 + [C_u K_u\ 0\ 0]\tilde{V}(k) + [W_e\ W_e\ W_{fuel}]x(k) + \tilde{C}_e(k)^T P_{load}(k)$$

The terms $C_x M_2 x(k)$, $C_x M_3$, $[W_e\ W_e\ W_{fuel}]x(k)$ *and* $\tilde{C}_e(k)^T P_{load}(k)$ can be removed from this optimisation since they are independent of the optimisation variables. Therefore, the final optimisation is

$$\min\ [1\ \cdots 1]\tilde{p}(k) + C_x M_1 \tilde{V}(k) + [C_u K_u\ 0\ 0]\tilde{V}(k)$$

$$= \min\ \underbrace{[(1\ \cdots 1)\ (C_x M_1 + [C_u K_u\ 0\ 0])]}_{S_\rho} \underbrace{[\tilde{\rho}(k)\ \tilde{V}(k)]^T}_{\tilde{V}_\rho}$$

Now we need to group the constraints together to complete the definition of our MILP optimisation problem. The constraints we have are

$$F_1 \tilde{V}(k) \leq F_2 + F_3 x(k)$$

$$-\rho \leq C_{\delta 1}\hat{\delta}(k) + C_{\delta 2}\delta(k-1) \leq \rho$$

The second constraints can be re-written as

$$-\rho - C_{\delta 1} K_\delta \tilde{\delta}(k) \leq C_{\delta 2}\delta(k-1)$$

$$-\rho + C_{\delta 1} K_\delta \tilde{\delta}(k) \leq -C_{\delta 2}\delta(k-1)$$

Therefore,

$$\underbrace{\begin{bmatrix} 0 & F_1 \\ -I & [0\ -C_{\delta 1}K_\delta\ 0] \\ -I & [0\ C_{\delta 1}K_\delta\ 0] \end{bmatrix}}_{F_{\rho,1}} \tilde{V}_\rho(k) \leq \underbrace{\begin{bmatrix} F_2 + F_3 x(k) \\ C_{\delta 2}\delta(k-1) \\ -C_{\delta 2}\delta(k-1) \end{bmatrix}}_{F_{\rho,2}}$$

Therefore our optimisation problem is

$$\min\ S_\rho \tilde{V}_\rho(k)$$

subject to

$$F_{\rho,1}\tilde{V}_\rho(k) \leq F_{\rho,2}$$

This has to be translated and optimised on MATLAB. We used the Gurobi solver for this purpose. The code is attached in Appendix D of this report. The simulations and their results are shown in the next section.

2.9 Closed-Loop Simulation of the Hybrid System

In the previous section, we computed the cost function for just one simulation time step. In this section we close the loop. This means that in each time step, we use the receding horizon principle and apply only the first optimal MPC control input.Then we feed this back into the system for the next time step. Since our sampling time is 15 minutes or 0.25 hrs, we choose our simulation time to be 48 hours or $48/T_s = 192$ time steps.

Inside a loop that runs from the start to the end of the simulation time, the MILP model is computed at each time step with its corresponding constraints. Then the optimisation matrices c, A and b are defined as required by the Gurobi optimiser format and the optimal matrix $\tilde{V}_\rho(k)$ is computed for this time step. The

values in $\tilde{V}_\rho(k)$ are split up into its ρ, u, δ and z values. The first of these optimal u, δ and z values are then used to calculate the state $x(k+1)$ at the next time step, and the corresponding optimal cost, which now includes the terms omitted during the optimisation ($C_x M_2 x(k)$, $C_x M_3$, $[W_e\ W_e\ W_{fuel}]x(k)$ *and* $\tilde{C}_e(k)^T P_{load}(k)$) as constants. The results of these simulations are as presented below.

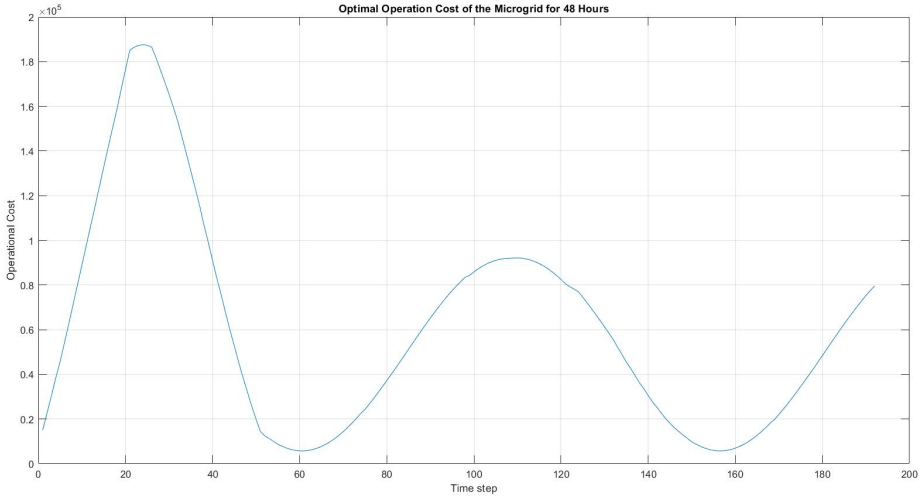


Figure 6: Optimal Operation Cost of the Microgrid(J) for 48 Hours

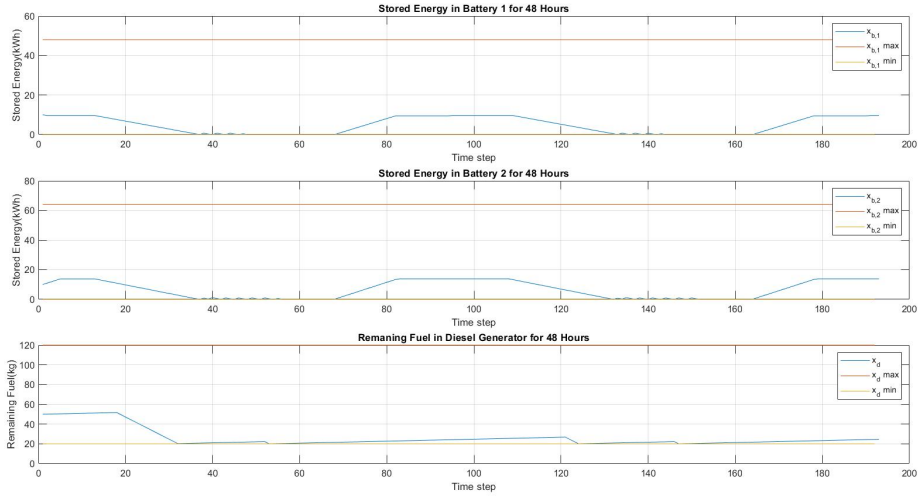


Figure 7: Variation of the States(x) of the system during 48 Hours

The optimal cost has been calculated for each time step and plotted in figure 6. The sinusoidal form of the cost can come as no surprise considering the sinusoidal waveform of the $\tilde{C}_e$ function. Initially, the controller has some difficulty in reducing the cost. At that time, just the batteries are not sufficient in order to do this. So the generator (although changing its state and using the generator are both penalised heavily) is called into action and it generates sufficient power to reduce the values of P_{imp} enough to bring

it back in control of the lightly penalised batteries again. This can be seen from the graph of the states and the inputs in figures 7 and 8, respectively. After this point, the generator is not used much, partially because the fuel it has is not enough for heavy operation, and also because it is expensive to use the generator.

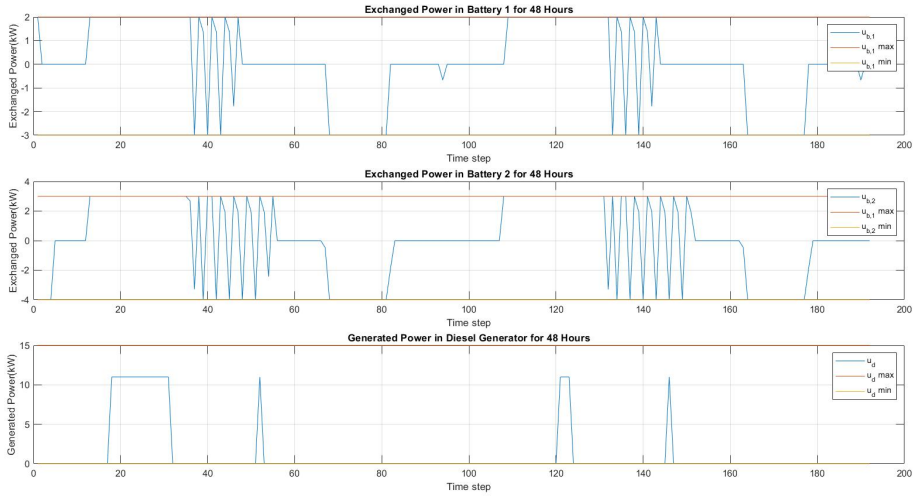


Figure 8: Variation of the optimal inputs(u) of the system during 48 Hours

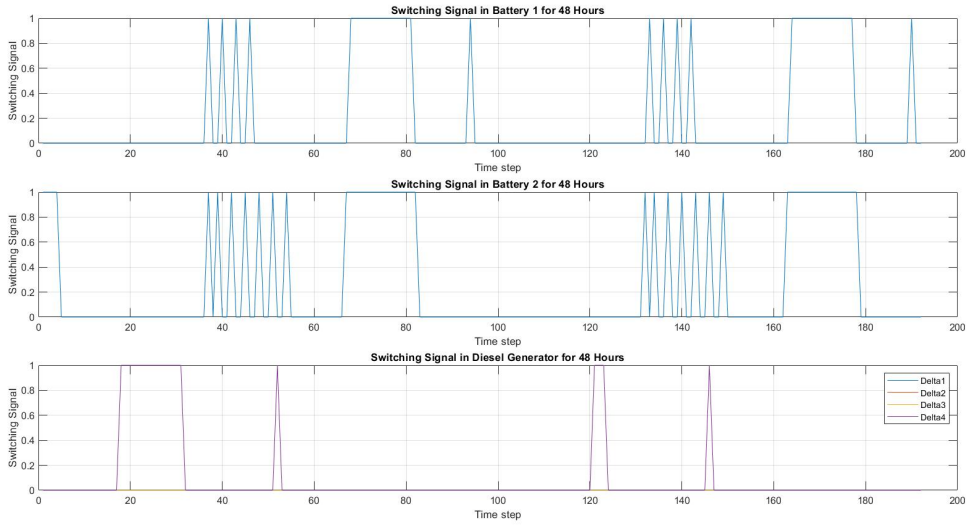


Figure 9: Variation of the Switching Signals(δ) of the system during 48 Hours

From Figure 9 it can be observed that only δ_4 seems to be triggered. This paints a picture that the generator is only used in extreme cases to generate power and contribute to the reduction in the cost. Coming to the batteries, they are consistently used throughout the 48 hours. Furthermore, they seem to be charging and discharging at similar times. They charge when the cost is low. This makes sense, since at that point the

demand is low and so the supply can be low too. When the demand (cost) rises in a sinusoidal pattern, so does the use of the batteries.

2.10 Closed-Loop Hybrid System with Additional battery constraint of 20% charge

In this section an additional requirement is imposed that at every 12^{th} hour (i.e. 12AM and 12PM), it needs to be ensured that the charging of the batteries is at least 20% of their maximum value. This means for Battery 1, the charge should be at least 9.6kWh at 12AM and 12PM and for Battery 2 the charge should be at least 12.8kWh. In order to simulate this, it is assumed that at t=0 the simulation is at 12PM or 12AM. At that time these constraints is ignored, since the initial charge levels are already provided in the question. Since our simulation is of a length of 48 hours, with a time step of 0.25 hrs, it can be computed that each 48 steps (12 hours divided by 0.25 hours), the simulation is either at 12PM or 12AM. So during the simulation such a point will be reached at least four times. The reason why it is at least 4 is because of the planning horizon, which should also consider this constraint in its future time steps. What this means is that if it is currently 11:15:00, then three steps into the future this state constraint needs to be incorporated. So if the planning horizon spanning greater than three time steps, this additional constraint need to be accounted for.

In order to do this, at each simulation step, corresponding to each value of k+j, the code searches for the k+j value which is a multiple of 48. Since the prediction horizon is 25 (which is quite a lot smaller than 48), there will not be such an occurrence in most of the cases. Then the matrices and the optimisation are equal to the ones performed in 2.9. However, in the case that 12AM or 12PM is the current iteration or a part of the planning horizon, some modifications need to be made to the state matrices. More specifically, at the location of the 48^{th} time step, we need to impose the constraint that the battery state at that instant should be greater than or equal to 20% of the maximum charge. This imposition is only on the battery state. Nothing needs to be done with the diesel generator. This additional constraint on the state will involve numerical and dimensional modifications to $\hat{E}_1$ and $\hat{g}_5$ and just dimensional changes to $\hat{E}_2, \hat{E}_3, \hat{E}_4$. Lets say that the 48th time step is at k+τ. Corresponding to this, our modified constraints are now,

$$\underbrace{\begin{bmatrix} E_1 & 0 & \dots & 0 & 0 \\ 0 & E_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & E_1 & 0 \\ 0 & 0 & \dots & 0 & I \\ 0 & 0 & \dots & 0 & -I \\ 0 & 0 & -I & 0 & 0 \end{bmatrix}}_{\hat{E}_1} \begin{bmatrix} x(k) \\ x(k+1) \\ \vdots \\ x(k+\tau) \\ \vdots \\ x\left(k+N_p-1\right) \\ x\left(k+N_p\right) \end{bmatrix} + \underbrace{\begin{bmatrix} E_2 & 0 & \dots & 0 \\ 0 & E_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_2 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_2} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+\tau) \\ \vdots \\ u\left(k+N_p-1\right) \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} E_3 & 0 & \dots & 0 \\ 0 & E_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_3 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_3} \begin{bmatrix} \delta(k) \\ \delta(k+1) \\ \vdots \\ \delta(k+\tau) \\ \vdots \\ \delta\left(k+N_p-1\right) \end{bmatrix} + \underbrace{\begin{bmatrix} E_4 & 0 & \dots & 0 \\ 0 & E_4 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & E_4 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\hat{E}_4} \begin{bmatrix} z(k) \\ z(k+1) \\ \vdots \\ z(k+\tau) \\ \vdots \\ z\left(k+N_p-1\right) \end{bmatrix} \leq \underbrace{\begin{bmatrix} g_5 \\ g_5 \\ \vdots \\ g_5 \\ \bar{x} \\ -\underline{x} \\ -x_{12} \end{bmatrix}}_{\hat{g}_5}$$

where $x_{12} = [0.2\bar{x}_{b,1}\ 0.2\bar{x}_{b,2}\ \underline{x}_d]^T$. The position of the identity matrix in $\hat{E}_1$ should correspond to the

term which represents 12AM or 12PM (i.e. the $(k+\tau)$th term). Everything else remains the same and the simulation can be done from here on exactly like it was done previously. The code is attached in Appendix D.1 and E of this report. The results are as shown below.

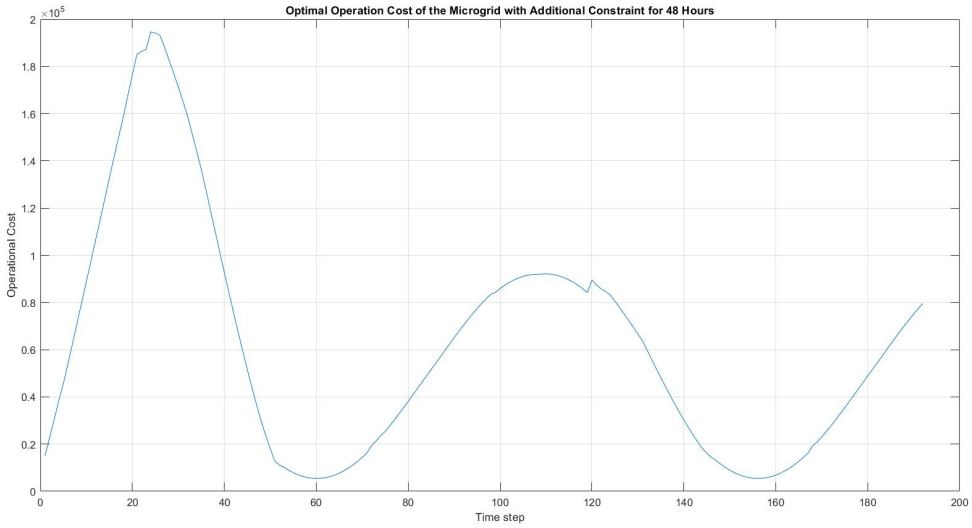


Figure 10: Optimal Operation Cost of the Microgrid(J) with 20% Charge Constraint for 48 Hours

Logically it can be expected that imposing such a constraint on states would increase the cost. Because an extra constraint means less freedom during to optimisation and thus a less optimal solution. Indeed, that is what happens, as can be seen in figure 10. Although the general pattern is identical to the cost variation in figure 6, the amplitude is larger in this case.

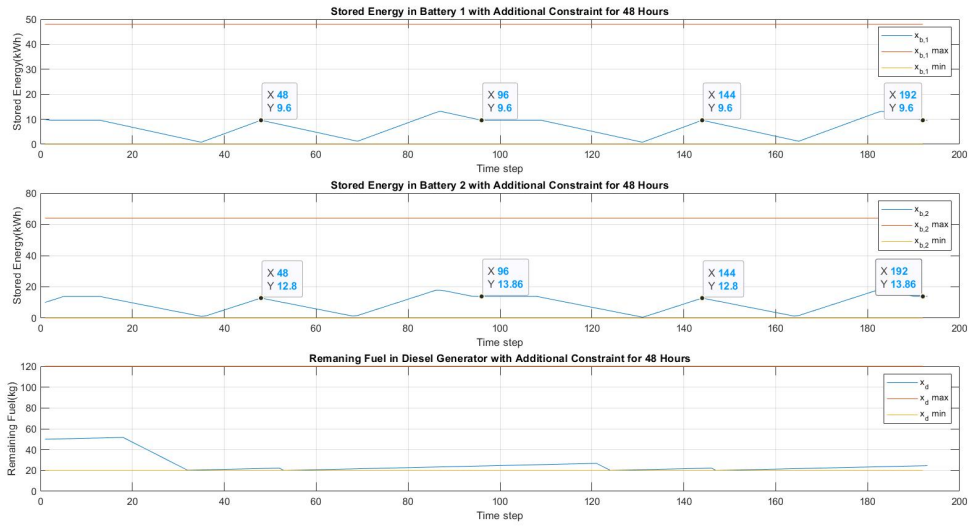


Figure 11: Variation of the States(x) with 20% Charge Constraint of the system during 48 Hours

In order to confirm that at 12AM, and 12PM the battery states are indeed at least at 20% charge, the points have been marked in the states variation diagram of figure 11. Keep in mind that the charge in Battery 1 should be at least 9.6kWh and the charge in Battery 2 should be at least 12.8kWh. Both these constraints are satisfied. Referring to the actions that take place (Figure 12), it can be observed that the fluctuations in exchanged power are smoother than what was observed in figure 8. In Section 2.9 there were time intervals where the battery had no charge, but here the difference seems to be that as soon as the battery charge dips, it is charged immediately. A reason that the batteries 'know' that they have to be charged, is the additional constraint applied in the horizon.

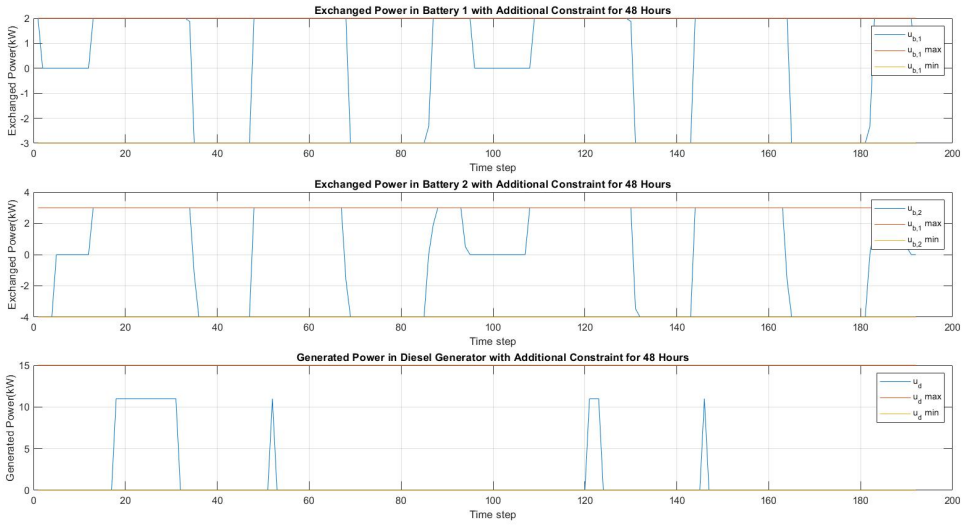


Figure 12: Variation of the optimal inputs(u) with 20% Charge Constraint of the system during 48 Hours

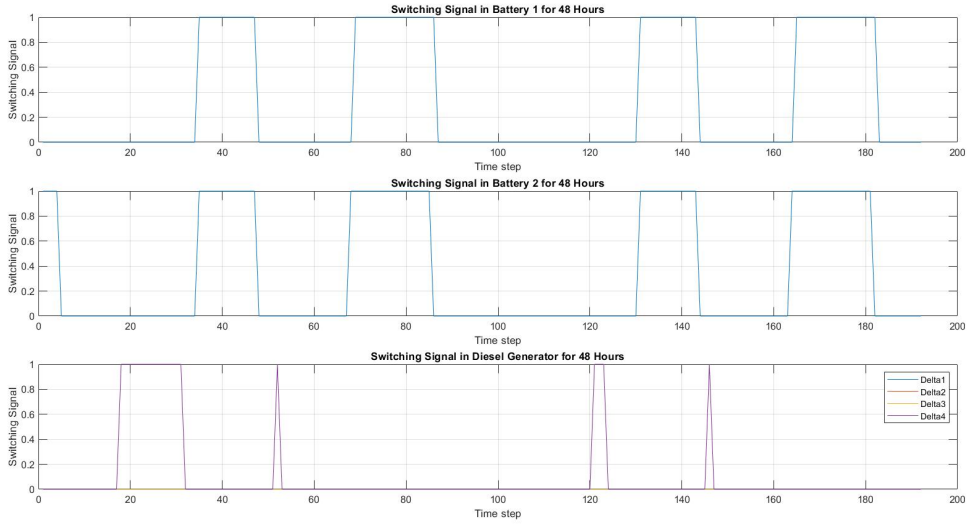


Figure 13: Variation of the Switching Signals(δ) of the system with additional constraint during 48 Hours

Just like last time, both batteries are very much synchronised, as we can see with the charging-discharging δ patterns in figure 13. Just like in the case of the control inputs, the waveform here is also very smooth just like in figure 9 from the previous section.

3 Evaluation and Conclusions

In this assignment we modelled a hybrid automaton from a system in our surroundings and then we simulated a model of a microgrid connected to the power grid. We were able to simulate this model both with and without the 12AM/PM constraint, where we got the results that satisfied the constraints we had defined. Of course not everything worked immediately, which gave us a few different insights.

Part 1 of this assignment deals with choosing a hybrid system from our daily lives, and attempting to mathematically represent it as a Hybrid Automaton. We chose a T-Junction with traffic lights in Delft for this purpose. We provided a mathematical description of the system, its variables and constraints. We then quantified the states and the transitions. Once this was done, we next modelled the system as a hybrid automaton with the above mentioned states, transitions and conditions. For part 1 the main challenge was to translate the intersection into an automaton that was probable as well as manageable. This is why we used our own experience with this intersection as a guide to make sure that the values we chose for the guards, invariant sets and dynamics represented the actual traffic situation.

Part 2 represents the major part of this assignment. The system we needed to describe was a micro-grid system with batteries and generators. We need to design an efficient energy management system to ensure that the appropriate energy demands are met at minimal cost incurred. We will outline the objective and tasks part by part.

We first created a PWA model of a battery with certain specifications given. From this PWA model, we constructed a MLD model of the battery, which involved mathematically representing logical states and using additional mixed-logical states. We defined the state update equation and the constraint equations. For the diesel generator we were given a non-linear model of the fuel consumption and had the task of fitting it into four linear PWA functions. After that, we had to go through the same process as for the batteries. This time there many more constraint equations in particular, owing to the number of PWA functions in the model, and with this the number of Mixed Logical, and Logical variables.

Now that we had a MLD model for a battery and a generator, the next task was to combine the two, thus creating a larger combined MLD Hybrid system containing the battery and generator equations and constraints together. This system was also coded in MATLAB. Next, we were provided a cost function for the microgrid, and the task was to use the MPC concept to compute the optimal control input values for a timestep. However, the cost function provided in its raw form was not ideal for a MILP optimisation. Several irrelevant terms needed to be omitted, some terms needed to be brought into a linear form and we had to group the optimisation variables together. This same operation had to be performed on the constraint equations. Now all that was left to do, was to close the loop over this optimisation (i.e. repeatedly compute the optimal control sequences, feed them to the system, use the new state to recompute the optimal sequences and so on). The optimisation was successfully performed using the Gurobi Optimiser Interface on MATLAB. The final part of this assignment was to add an additional constraint on the state of the batteries at a certain time instant, and study its effect on the optimisation. This too was successfully implemented and an increase in cost and change in behaviour of the system were observed, compared to the previously obtained results. Thus, the hybrid energy management system with an optimal MPC based controller to minimise the operational cost was successfully implemented through this assignment.

While working through part 2 of the assignment, the first thing we learned, was that it was important to first do all the computations by hand. Only when we got the equation in the right form, we would start to implement it in MATLAB. Especially, transforming the problem to the MILP model of MPC problem seemed complex at first. However, by first figuring out what the form should be in the end and working it out step by step, we were able to get there. This made it possible for the MATLAB code to compute the result rather efficiently.

Another related aspect, is that the matrices would get quite large. For example, when combining the battery and generator matrices or even more so when transforming the system into the MPC format. This made it easy to make mistakes during the computation by hand or when transferring the matrices into MATLAB. This made us realise that it was extra important to write down intermediate results. This made it easier to check where we had gone wrong. This also resulted in another advantage, because at first we were using two very similar functions for 2.8 & 2.9 and for 2.10. There was a mistake somewhere in our code for 2.9. This made us realise that with a small alteration, we could be using the same function for all three questions and even completely the same script for both 2.9 and 2.10. This resulted in a more efficient MATLAB code (and a smaller Appendix).

Next to that, when we first simulated 2.9, we noticed that the input u_d of the diesel generator was not coupled correctly to the δ values. It was possible for the input to be larger than zero, while all δs where zero. Of course this should not be the case. This made us realise, that apart from maximising the sum of all δ values to be smaller or equal to one, the input should also by smaller than this sum multiplied with $\bar{u}_d$ ($u(k) \leq (\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k))\bar{u}_d$).

A suggestion for other people doing the same assignment would be to also look into the other two toolbox suggestions given in the assignment (glpx and cplex). We only looked at the Gurobi optimiser, because we already had experience with this toolbox and we were able to make to code work using this. So it might be worth it to also read up on the other two before making a choice.

To summarise, this assignment was pivotal in enhancing and realising the theory covered during the lectures of the course. Through this assignment, we learnt translation of real-world hybrid systems into mathematical format. Furthermore, we learnt how to model hybrid systems and a lot of deeper mathematical insight was gained into optimisation methods and tools. Moreover, we learnt how an MPC problem can be represented and solved as a MILP optimisation. This assignment was indeed a good representation of Modelling and Control of Hybrid Systems.

Appendix

A Question 2.3

```
%Vivek Varma and Justine Kroese
clc;
clear all;
close all;
%% 2.3
syms u a1 b1 a2 b2 a3 b3 a4 b4;
u1=5;
u2=6.5;
u3=11;
%Non-linear functions
f1=(u^2)+4;
f2=4*u;
f3=(-9.44*u^3)+(166.06*u^2)-(948.22*u)+1790.28;
f4=-11.78*u+132.44;
f5=(4.01*u^2)-(20.94*4.01*u)+17.79+(4.01*10.47^2);
%Optimisation objective function
fun=@(x) int((f1-(x(1)+x(2)*u))^2,0, 2)+int((f2-(x(1)+x(2)*u))^2, 2, 5)+ ...
int((f3-(x(3)+x(4)*u))^2, 5, 6.5)+int((f3-(x(5)+x(6)*u))^2, 6.5, 7)+ ...
int((f4-(x(5)+x(6)*u))^2, 7, 9)+int((f5-(x(5)+x(6)*u))^2, 9, 11)+int((f5-(x(7)+x(8)*u))^2, 11,15);
guess = [1 1 1 1 1 1 1 1];
options = optimoptions(@ga,'PopulationSize', 700, 'MaxGenerations', 1600,'EliteCount', 25, ...
'Display', 'iter');
[goptidiscrete, optvaldiscrete] =ga(fun, 8, [], [], [], [], [], [], [], [], options);
%% 2.3 Plots
pc=piecewise(0<=u & u<2, (u^2)+4, 2<=u & u<5, 4*u, 5<=u & u < 7, (-9.44*u^3)+(166.06*u^2)- ...
(948.22*u)+1790.28, 7<=u & u< 9, -11.78*u+132.44, 9<=u & u<=15, (4.01*u^2)-(20.94*4.01*u)+ ...
17.79+(4.01*10.47^2));
figure
fplot(pc, [0 15])
hold on;
grid on;
pc2=piecewise(0<=u & u<5, goptidiscrete(1)+goptidiscrete(2)*u, 5<=u & u<6.5,goptidiscrete(3)+ ...
goptidiscrete(4)*u, 6.5<=u & u < 11, goptidiscrete(5)+goptidiscrete(6)*u, 11<=u & u<= 15, ...
goptidiscrete(7)+goptidiscrete(8)*u);
fplot(pc2, [0 15])
xlabel('$u_d(k)$', 'Interpreter','latex')
ylabel('$f(u_d(k))/\hat{f}(u_d(k))$', 'Interpreter','latex')
legend('$f(u_d(k))$','$\hat{f}(u_d(k))$', 'Interpreter','latex')
title('Non-linear model vs PWA model of Diesel Generator')
```

B Question 2.4

```
%% 2.4
%Optimisation objective function
fun=@(x) int((pc-piecewise(0<=u & u<x(3), x(1)+x(2)*u, x(3)<=u & u<x(6), x(4)+x(5)*u, ...
x(6)<=u & u<x(9), x(7)+x(8)*u, x(9)<=u & u<=15, x(10)+x(11)*u))^2, 0, 15)
%Initial guess=Solution of optisation in 2.3
x0=[1.6317452147803 3.5391964828094 5 -85.6286320070643 20.8671127088603 6.5 111.8700873269495 ...
-9.1652452629377 11 -207.2789168622208 19.6967714691146];
options = optimoptions(@simulannealbnd,'Display', 'iter');
%Since the values of u1, u2, u3 have to be bounded
lb=[-Inf; -Inf; 0; -Inf; -Inf; 0; -Inf;-Inf;0;-Inf;-Inf];
```

```
ub=[Inf; Inf; 15; Inf; Inf; 15; Inf;Inf;15;Inf;Inf];
[gopti, optval]=simulannealbnd(fun,x0,lb,ub,options)
%% 2.4 Plots
pc=piecewise(0<=u & u<2, (u^2)+4, 2<=u & u<5, 4*u, 5<=u & u < 7, (-9.44*u^3)+(166.06*u^2)- ...
(948.22*u)+1790.28, 7<=u & u< 9, -11.78*u+132.44, 9<=u & u<=15, (4.01*u^2)-(20.94*4.01*u)+ ...
17.79+(4.01*10.47^2));
figure
fplot(pc, [0 15])
hold on;
grid on;
pc2=piecewise(0<=u & u<gopti(3), gopti(1)+gopti(2)*u, gopti(3)<=u & u<gopti(6), ...
gopti(4)+gopti(5)*u, gopti(6)<=u & u < gopti(9), gopti(7)+gopti(8)*u, gopti(9)<=u & u<= 15, ...
gopti(10)+gopti(11)*u);
fplot(pc2, [0 15])
xlabel('$u_d(k)$', 'Interpreter','latex')
ylabel('$f(u_d(k))/\hat{f}(u_d(k))$', 'Interpreter','latex')
legend('$f(u_d(k))$','$\hat{f}(u_d(k))$', 'Interpreter','latex')
title('Non-linear model vs PWA model of Diesel Generator with variable PWA limits')
```

C Question 2.7

```
%% Exercise 2.7
%% Variables
Ts=0.25;                % sampling time [h]

% Specification batteries
bat1.eta_c=0.9;         % charging efficiency battery 1
bat2.eta_c=0.95;        % charging efficiency battery 2
bat1.eta_d=0.8;         % discharging efficiency battery 1
bat2.eta_d=0.77;        % dicharging efficiency batttery 2
bat1.x_max=48;          % max stored energy battery 1 [kWh]
bat2.x_max=64;          % max stored energy battery 2 [kWh]
bat1.u_min=-3;
bat2.u_min=-4;
bat1.u_max=2;
bat2.u_max=3;

% Specification diesel generator
dies.x_min=20;          % [kg]
dies.x_max=120;         % [kg]
dies.u_max=15;          % [kW]
dies.Rf=0.4;            % [kg/h]


a1=1.6318;
a2=-85.6286;
a3=111.8701 ;
a4=-207.2789;
b1=3.5292;
b2=20.8671;
b3=-9.1653;
b4=19.6968;
dies.a=[a1 a2 a3 a4];
dies.b=[b1 b2 b3 b4];
u1=5;
u2=6.5;
u3=11;

%% Description of the MLD matrices

```

```
% Build MLD matrices batteries
MLDbat1=build_matrices_bat(bat1,Ts);
MLDbat2=build_matrices_bat(bat2,Ts);

% Build MLD matrices diesel generator
MLDdies.A=1;
MLDdies.B1=0;
MLDdies.B2=-Ts*dies.a;
MLDdies.B3=-Ts*dies.b;
MLDdies.B4=dies.Rf*Ts;
MLDdies.E1=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 -1]';
MLDdies.E2=[-1 1 -1 1 -1 1 -1 1 1 0 0 0 -1 1 0 0 -1 1 0 0 -1 1 0 0 -1 1 0 0]';
MLDdies.E3=[0 0 0 0; -u1+dies.u_max+eps 0 0 0;0 u1 0 0; 0 -u2+dies.u_max+eps 0 0;0 0 u2 0;...
0 0 -u3+dies.u_max+eps 0;0 0 0 u3; 0 0 0 0;-dies.u_max -dies.u_max -dies.u_max -dies.u_max;...
1 1 1 1; -u1 0 0 0;0 0 0 0; 0 0 0 0;dies.u_max 0 0 0; 0 -u2 0 0; 0 u1 0 0;0 0 0 0;...
0 dies.u_max 0 0;0 0 -u3 0; 0 0 u2 0;0 0 0 0;0 0 dies.u_max 0; 0 0 0 -15;0 0 0 u3;0 0 0 0;...
0 0 0 dies.u_max;0 0 0 0;0 0 0 0];
MLDdies.E4=[0 0 0 0; 0 0 0 0;0 0 0 0; 0 0 0 0;0 0 0 0; 0 0 0 0;0 0 0 0; 0 0 0 0;0 0 0 0;0 0 0 0;...
1 0 0 0; -1 0 0 0; 1 0 0 0;-1 0 0 0; 0 1 0 0; 0 -1 0 0;0 1 0 0; 0 -1 0 0;0 0 1 0; 0 0 -1 0;...
0 0 1 0;0 0 -1 0; 0 0 0 1;0 0 0 -1;0 0 0 1; 0 0 0 -1;0 0 0 0;0 0 0 0 ];
MLDdies.g5=[0 dies.u_max 0 dies.u_max 0 dies.u_max 0  15 0 1 0 0 0 dies.u_max 0 0 0 dies.u_max ...
0 1 0 dies.u_max 0 0 0 dies.u_max dies.x_max -dies.x_min]';


% Combining the matrices
sys.A=blkdiag(MLDbat1.A,MLDbat2.A,MLDdies.A);
sys.B1=blkdiag(MLDbat1.B1,MLDbat2.B1,MLDdies.B1);
sys.B2=blkdiag(MLDbat1.B2,MLDbat2.B2,MLDdies.B2);
sys.B3=blkdiag(MLDbat1.B3,MLDbat2.B3,MLDdies.B3);
sys.B4=[MLDbat1.B4;MLDbat2.B4;MLDdies.B4];

sys.E1=blkdiag(MLDbat1.E1,MLDbat2.E1,MLDdies.E1);
sys.E2=blkdiag(MLDbat1.E2,MLDbat2.E2,MLDdies.E2);
sys.E3=blkdiag(MLDbat1.E3,MLDbat2.E3,MLDdies.E3);
sys.E4=blkdiag(MLDbat1.E4,MLDbat2.E4,MLDdies.E4);
sys.g5=[MLDbat1.g5;MLDbat2.g5;MLDdies.g5];
```

C.1 Function to calculate battery matrices

```
function f=build_matrices_bat(bat,Ts)
A=1;
B1=-bat.eta_d*Ts;
B2=0;
B3=(bat.eta_d-bat.eta_c)*Ts;
B4=0;

E1=[0 0 0 0 0 0 1 -1]';
E2=[-1 1 0 0 -1 1 0 0]';
E3=[bat.u_min-eps bat.u_max -bat.u_max bat.u_min -bat.u_min bat.u_max 0 0]';
E4=[0 0 1 -1 1 -1 0 0]';
g5=[-eps bat.u_max 0 0 -bat.u_min bat.u_max bat.x_max 0]';

% Matrices are returned as a structure
f = struct('A',A,'B1',B1,'B2',B2,'B3',B3,'B4',B4,'E1',E1,'E2',E2,'E3',E3,'E4',E4,'g5',g5);
end
```

D Question 2.8

```
%% Exercise 2.8
%% Variables
Nb=2;                                           % amount of batteries
Np=25;                                          % prediction horizon
Nc=25;                                          % control horizon

Nx=size(sys.A,2);
Nu=size(sys.B1,2);
Ndelta=size(sys.B2,2);
Nz=size(sys.B3,2);

Nrho=(Nb+1)*Np;

x0=[10;10;50];
xk=x0;
delta_min1=[0;0;0;0;0;0];
deltak_min1=delta_min1;

% weight factors
weight.Wb1=3;                                   % weight factor batteries
weight.Wb2=4;
weight.Wd=10;                                   % weight factor diesel generator
weight.W_fuel=4;                                % weight factor fuel
weight.We=0.4;

k=0:1:Np-1;
weight.Ce=50+50*sin(pi*Ts*k/12)';               % benefit for importing energy
weight.P_load=zeros(Np,1);
for i=1:Np-1
    if i>20 && i<51
        weight.P_load(i)=30+2*i;
    elseif i>=51
        weight.P_load(i)=45;
    end
end

MILPsys=MILP_model(sys,bat1,bat2,dies,Nb,Np,Nc,weight,xk,deltak_min1,[]);

% solve the problem for the initial conditions
model.obj=MILPsys.c;
model.A=MILPsys.A;
model.rhs=MILPsys.b;
model.modelsense ='min';
model.vtype=repmat('C',size(MILPsys.c,2),1);
model.vtype(MILPsys.bin_part)='B';
model.lb =-inf(size(MILPsys.c,2),1);

optimum=gurobi(model);
opt.V=optimum.x;
opt.rho=opt.V(1:Nrho);
opt.u=opt.V(Nrho+1:(Nrho+Nu*Nc));
opt.delta=opt.V((Nrho+Nu*Nc+1):(Nrho+(Nu+Ndelta)*Nc));
opt.z=opt.V((Nrho+(Nu+Ndelta)*Nc+1):end);
```

D.1 Function to determine MILP description

```
function [MILPsys,Cx,M2,M3]=MILP_model(sys,bat1,bat2,dies,Nb,Np,Nc,weight,xk,deltak_min1,T12h)

Nx=size(sys.A,2);
Nu=size(sys.B1,2);
Ndelta=size(sys.B2,2);
Nz=size(sys.B3,2);

M2=[];                                                          %M2
for i=1:Np
    M2=[M2;sys.A^i];
end

M3=[];                                                          %M3
sum=0;
for i=1:Np
    sum=sum+sys.A^(i-1)*sys.B4;
    M3=[M3; sum];
end

T1=[];
T2=[];
T3=[];
matrt1=[];
matrt2=[];
matrt3=[];
for i=1:Np
    matrt1=[sys.A^(Np-1)*sys.B1 matrt1];
    T1=[T1;matrt1 zeros(size(sys.B1, 1), size(sys.B1, 2)*(Np-i))];      %T1
    matrt2=[sys.A^(Np-1)*sys.B2 matrt2];
    T2=[T2;matrt2 zeros(size(sys.B2, 1), size(sys.B2, 2)*(Np-i))];      %T2
    matrt3=[sys.A^(Np-1)*sys.B3 matrt3];
    T3=[T3;matrt3 zeros(size(sys.B3, 1), size(sys.B3, 2)*(Np-i))];      %T3
end

Inu=eye(Nx);
Ku=[];                                                          %Ku
for i=1:Nc
    Ku=blkdiag(Ku, Inu);
end
for j=1:(Np-Nc)
    Ku=[Ku; zeros(Nx, Nx*(Nc-1)) Inu];
end

Indelta=eye(Ndelta);
Kdelta=[];                                                      %Kdelta
for i=1:Nc
    Kdelta=blkdiag(Kdelta, Indelta);
end
for j=1:(Np-Nc)
    Kdelta=[Kdelta; zeros(Ndelta, Ndelta*(Nc-1)) Indelta];
end

Inz=eye(Nz);
Kz=[];                                                          %Kz
for i=1:Nc
    Kz=blkdiag(Kz, Inz);
end
for j=1:(Np-Nc)
    Kz=[Kz; zeros(Nz, Nz*(Nc-1)) Inz];
end

```

```
M1=[T1*Ku T2*Kdelta T3*Kz];                                          %M1

E1hat=[];
E2hat=[];
E3hat=[];
E4hat=[];
for i=1:Np
    E1hat=blkdiag(E1hat, sys.E1);
    E2hat=blkdiag(E2hat, sys.E2);
    E3hat=blkdiag(E3hat, sys.E3);
    E4hat=blkdiag(E4hat, sys.E4);
end

E1hat=blkdiag(E1hat, eye(3));
E1hat=[E1hat; zeros(3, 3*Np) -eye(3)];                               %E1hat


E2hat=[E2hat; zeros(6, size(E2hat, 2))];                             %E2hat
E3hat=[E3hat; zeros(6, size(E3hat, 2))];                             %E3hat
E4hat=[E4hat; zeros(6, size(E4hat, 2))];                             %E4hat

% constraint for 12 AM/PM
if ~isempty(T12h)==1
    E1hat=[E1hat;zeros(Nx,Nx*(T12h-1)) -eye(Nx) zeros(Nx,Nx*(Np-T12h)) zeros(Nx,Nx)];
    E2hat=[E2hat;zeros(Nx,Nu*Np)];
    E3hat=[E3hat;zeros(Nx,Ndelta*Np)];
    E4hat=[E4hat;zeros(Nx,Nz*Np)];
end

g5hat=[];
for i=1:Np
    g5hat=[g5hat; sys.g5];
end
xupper=[bat1.x_max; bat2.x_max; dies.x_max];
xlower=[0; 0; dies.x_min];

% constraint for 12 AM/PM
g5hat=[g5hat; xupper; -xlower];                                      %g5hat
if ~isempty(T12h)==1
    g5hat=[g5hat;-0.2*bat1.x_max; -0.2*bat2.x_max; -dies.x_min];
end

F1=E1hat*[zeros(Nx, size(M1, 2)); M1]+[E2hat*Ku E3hat*Kdelta E4hat*Kz]; %F1
F2=g5hat-E1hat*[zeros(Nx,1); M3];                                    %F2
F3=-E1hat*[eye(Nx); M2];                                             %F3

% Cost function
Cu_block=[-1,-1,-1];                                        % block that's repeated in Cu
Cu_block_rep=repmat(Cu_block, 1, Np);                       % repeat block Np times
Cu_cellarray=mat2cell(Cu_block_rep, size(Cu_block,1), repmat(size(Cu_block,2),1,Np)); % makes ...
% cell array from repeated block
Cu=weight.Ce'*blkdiag(Cu_cellarray{:});                     % block diagonal matrix * weights

Cx=[zeros(1,(Np-1)*Nx),weight.We,weight.We,weight.W_fuel];           % Cx

Wdelta=[weight.Wb1 0 0 0 0 0; 0 weight.Wb2 0 0 0 0; 0 0 weight.Wd  weight.Wd  weight.Wd  weight.Wd];
% Wdelta
Cdelta1=zeros(size(Wdelta,1)*Np,size(Wdelta,2)*Np);
Cdelta1((size(Cdelta1,1)-size(Wdelta,1)+1):end,(size(Cdelta1,2)-size(Wdelta,2)+1):end)=Wdelta;
for i=1:Np-1
    Cdelta1(1+size(Wdelta,1)*(i-1):size(Wdelta,1)+size(Wdelta,1)*(i-1),...
    1+size(Wdelta,2)*(i-1):size(Wdelta,2)+size(Wdelta,2)*(i-1))=Wdelta;
    Cdelta1(size(Wdelta,1)+1+size(Wdelta,1)*(i-1):size(Wdelta,2)+size(Wdelta,1)*(i-1),...
    1+size(Wdelta,2)*(i-1):size(Wdelta,2)+size(Wdelta,2)*(i-1))=-Wdelta;
end
```

```
Cdelta2=[-Wdelta;zeros(size(Cdelta1,1)-size(Wdelta,1),size(Wdelta,2))];

% Final matrices
Nrho=(Nb+1)*Np;
Srho=[ones(1,Nrho),([Cu*Ku,zeros(1,Ndelta *Nc),zeros(1,Nz*Nc)]+Cx*M1)];
Frho1=[sparse(size(F1,1),Nrho) F1;-speye(Nrho) sparse(size(Cdelta1*Kdelta,1),Nu*Nc) ...
-Cdelta1*Kdelta sparse(size(Cdelta1*Kdelta,1),Nz*Nc);-speye(Nrho) ...
sparse(size(Cdelta1*Kdelta,1),Nu*Nc) Cdelta1*Kdelta sparse(size(Cdelta1*Kdelta,1),Nz*Nc)];
%needs to be sparse
Frho2=[F2+F3*xk;Cdelta2*deltak_min1;Cdelta2*deltak_min1];

MILPsys.c=Srho;
MILPsys.A=Frho1;
MILPsys.b=Frho2;
MILPsys.bin_part =(Nrho+Nu*Nc+1):(Nrho+(Nu+Ndelta)*Nc);
MILPsys.Cx=Cx;
end
```

E Question 2.9 & 2.10

```
%% Exercise 2.9 & 2.10
%% simulation time
N=48/Ts;                                                    % 48h is 192 steps

% initialisation
x192=zeros(3,Np+1);
u192=zeros(3,Np+1);
delta192=zeros(6,Np+1);
z192=zeros(6,Np+1);
x192(:,1)=[10;10;50];
constraints192=zeros(1,Np+1);
J192=zeros(1,Np);
T12h=0;

Question210=false;                                          % false for 2.9, true for 2.10

for j=1:N
    t=j+k;                  % for each step in the horizon we follow the whole prediction horizon
    weight.Ce=50+50*sin(pi*Ts*t/12)';
    weight.P_load=zeros(Np,1);

    weight.P_load(t<=20)=0;
    weight.P_load(t>=21&t<=50)=30+2*t(t>= 21&t<=50);
    weight.P_load(t>=51)=45;

    if Question210
        T12h=find(mod(t,48)==0);
    else
        T12h=[];
    end


    if j>1
        [MILPsys192,Cx,M2,M3]=MILP_model(sys,bat1,bat2,dies,Nb,Np,Nc,weight,x192(:,j),...
        delta192(:,j-1),T12h);
    else
        [MILPsys192,Cx,M2,M3]=MILP_model(sys,bat1,bat2,dies,Nb,Np,Nc,weight,x192(:,j),...
        delta_min1,T12h);
    end

    model=struct();
    model.obj=MILPsys192.c;
    model.A=MILPsys192.A;
    model.rhs=MILPsys192.b;
    model.modelsense ='min';
    model.vtype=repmat('C',size(MILPsys192.c,2),1);
    model.vtype(MILPsys192.bin_part)='B';
    model.lb =-inf(size(MILPsys192.c,2),1);                        % otherwise no negative values

    params.outputflag = 0;
    optimum192=gurobi(model,params);
    opt192.V=optimum192.x;
    opt192.rho=opt192.V(1:Nrho);
    opt192.u=opt192.V(Nrho+1:(Nrho+Nu*Nc));
    opt192.delta=opt192.V((Nrho+Nu*Nc+1):(Nrho+(Nu+Ndelta)*Nc));
    opt192.z=opt192.V((Nrho+(Nu+Ndelta)*Nc+1):end);

    u192(:,j)=opt192.u(1:Nu);
    delta192(:,j)=opt192.delta(1:Ndelta);
    z192(:,j)=opt192.z(1:Nz);

```

```
    x192(:,j+1) = sys.A*x192(:,j)+sys.B1*u192(:,j)+sys.B2*delta192(:,j)+sys.B3*z192(:,j)+sys.B4;

    % check constraints:
    constraints192(j)=all(sys.E1*x192(:,j)+sys.E2*u192(:,j)+sys.E3*delta192(:,j)+ ...
    sys.E4*z192(:,j)-sys.g5<=eps);

    J192(j)=optimum192.objval+Cx*M2*x192(:,j)+Cx*M3+[weight.We weight.We weight.W_fuel]*x192(:,j)+ ...
    weight.Ce'*weight.P_load;

end

figure
plot(J192)
grid on;
if Question210
    title('Optimal Operation Cost of the Microgrid with Additional Constraint for 48 Hours')
else title('Optimal Operation Cost of the Microgrid for 48 Hours')
end
xlabel('Time step')
ylabel('Operational Cost')

figure
subplot(3,1,1)
plot(x192(1,:))
hold on;
plot([1:N], ones(1, N)*bat1.x_max)
hold on;
plot([1:N], ones(1, N)*0)
grid on;
if Question210
    title('Stored Energy in Battery 1 with Additional Constraint for 48 Hours')
else title('Stored Energy in Battery 1 for 48 Hours')
end
xlabel('Time step')
ylabel('Stored Energy(kWh)')
legend('x_{b,1}', 'x_{b,1} max', 'x_{b,1} min')
subplot(3,1,2)
plot(x192(2,:))
hold on;
plot([1:N], ones(1, N)*bat2.x_max)
hold on;
plot([1:N], ones(1, N)*0)
grid on;
if Question210
    title('Stored Energy in Battery 2 with Additional Constraint for 48 Hours')
else title('Stored Energy in Battery 2 for 48 Hours')
end
xlabel('Time step')
ylabel('Stored Energy(kWh)')
legend('x_{b,2}', 'x_{b,2} max', 'x_{b,2} min')

subplot(3,1,3)
plot(x192(3,:))
hold on;
plot([1:N], ones(1, N)*dies.x_max)
hold on;
plot([1:N], ones(1, N)*dies.x_min)
grid on;
if Question210
    title('Remaining Fuel in Diesel Generator with Additional Constraint for 48 Hours')
else title('Remaining Fuel in Diesel Generator for 48 Hours')
end
xlabel('Time step')
ylabel('Remaining Fuel(kg)')
legend('x_{d}', 'x_{d} max', 'x_{d} min')
```

```
127
128 figure
129 subplot(3,1,1)
130 plot(u192(1,:))
131 hold on;
132 plot([1:N], ones(1, N)*bat1.u_max)
133 hold on;
134 plot([1:N], ones(1, N)*bat1.u_min)
135 grid on;
136 if Question210
137     title('Exchanged Power in Battery 1 with Additional Constraint for 48 Hours')
138 else title('Exchanged Power in Battery 1 for 48 Hours')
139 end
140 xlabel('Time step')
141 ylabel('Exchanged Power(kW)')
142 legend('u_{b,1}', 'u_{b,1} max', 'u_{b,1} min')
143 subplot(3,1,2)
144 plot(u192(2,:))
145 hold on;
146 plot([1:N], ones(1, N)*bat2.u_max)
147 hold on;
148 plot([1:N], ones(1, N)*bat2.u_min)
149 grid on;
150 if Question210
151     title('Exchanged Power in Battery 2 with Additional Constraint for 48 Hours')
152 else title('Exchanged Power in Battery 2 for 48 Hours')
153 end
154 xlabel('Time step')
155 ylabel('Exchanged Power(kW)')
156 legend('u_{b,2}', 'u_{b,1} max', 'u_{b,2} min')
157 subplot(3,1,3)
158 plot(u192(3,:))
159 hold on;
160 plot([1:N], ones(1, N)*dies.u_max)
161 hold on;
162 plot([1:N], ones(1, N)*0)
163 grid on;
164 if Question210
165     title('Generated Power in Diesel Generator with Additional Constraint for 48 Hours')
166 else title('Generated Power in Diesel Generator for 48 Hours')
167 end
168 xlabel('Time step')
169 ylabel('Generated Power(kW)')
170 legend('u_{d}', 'u_{d} max', 'u_{d} min')
171
172
173 figure
174 subplot(3,1,1)
175 plot(delta192(1,:))
176 grid on;
177 title('Switching Signal in Battery 1 for 48 Hours')
178 xlabel('Time step')
179 ylabel('Switching Signal')
180 subplot(3,1,2)
181 plot(delta192(2,:))
182 grid on;
183 title('Switching Signal in Battery 2 for 48 Hours')
184 xlabel('Time step')
185 ylabel('Switching Signal')
186 subplot(3,1,3)
187 plot(delta192(3:end,:)')
188 grid on;
189 title('Switching Signal in Diesel Generator for 48 Hours')
190 xlabel('Time step')
191 ylabel('Switching Signal')
```

```
192 legend('Delta1', 'Delta2' , 'Delta3', 'Delta4')
```