Delft University of Technology

Systems and Control

SC42075

Modeling and Control of Hybrid Systems Assignment

Authors:

Miranda van Duijn 4355776
Jessie van Dam 4395832

June 24, 2019

Introduction

This report contains the results and findings of the assignment that is part of the course SC42075 Modelling and Control of Hybrid Systems at Delft University of Technology. The goal of this assignment is twofold.

In the first part of the assignment, a self-chosen real-life system that can be considered as a hybrid system was described and represented as a hybrid automaton. The chosen system is a roller coaster in a theme park.

In the second part of the assignment, the energy management of microgrids is considered. The microgrid is connected to the main powergrid. The microgrid consists of several subsystems, being the diesel generator and its fuel tanks, two batteries and an energy management system. It is assumed that the energy management system has an accurate prediction of the load in the microgrid and that it is able to communicate without delay with the diesel generator and the two batteries. Furthermore, it is assumed that the electrical connection between the main power grid and the microgrid is not physically limited, meaning that the power balance can always be maintained. The power balance describes that the amount of power used by the two charging batteries and the load is equal to the power provided by the discharging batteries, the power grid and the diesel generator. The microgrid is represented in figure 3 below.

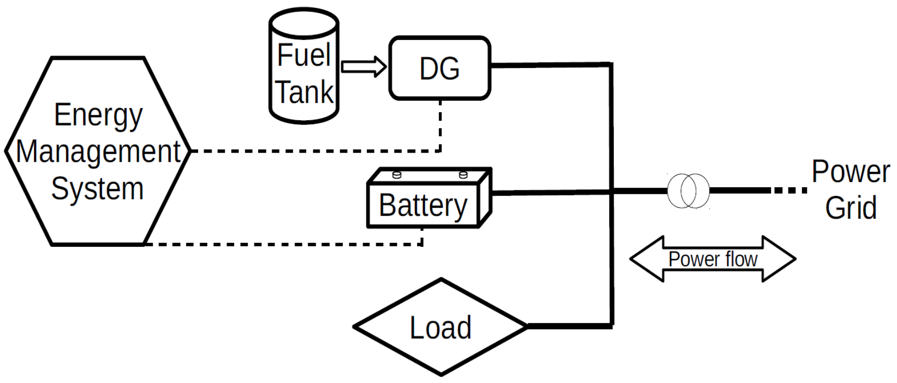

Figure 1: Schematic representation of the considered microgrid

The aim of this assignment is to minimize the operational cost of the microgrid, by designing a hybrid model predictive controller for the energy management system. A price is defined for the exchanged energy per kWh between the microgrid and the power grid. Hence, whenever electrical power is imported from the main power grid, additional operational costs arise because of the price of importing energy. On the other hand, revenue is made whenever energy is exported to the power grid. The energy management system is able to control the operation of the diesel generators and batteries and can in this way influence the costs that are made by the microgrid.

Part 1: Hybrid System Example

Step 1.1

A hybrid system that can be described by an automaton is the waiting line for the *Python*, a roller coaster located in the Dutch theme park *Efteling*.

Figure 2: The Python roller coaster

Variables
The dynamics of the system are defined by two variables $x(t)$ and $y(t)$. Here, $x(t)$ equals time [min] elapsed since a train has been placed or removed on the track, or since the initial condition. $y(t)$ represents the length of the waiting line [m]. Both variables are continuous. There is also one discrete variable, denoted by $P(t)$, which is the status of the Python. When $P(t) = on$, the Python is operational, meaning that visitors can ride the rollercoaster. When $P(t) = off$, the Python is non-operational, meaning that visitors cannot ride the rollercoaster. The second continuous variable, $y(t)$, is described by discontinuous dynamics which can be seen in the automaton in section 1.2.

Dynamics of the system
The initial state is $x(t) = 0, \quad y(t) = 0$ and when there is one train on the roller coaster. The initial state is in the first node. The waiting line length grows at the rate of $\dot{y} = a - b_1$ per second. This means the waiting line grows larger by constant a per time, because of people joining the waiting line to get on the roller coaster. The length of the line also decreases by constant b_1 per time, because people get on the train. In reality, the decrease of the waiting line is not continuous over time, because the waiting line only decreases when a train has finished the track of the roller coaster. However, for simplicity, the time average is taken as a continuous decrease.

When the waiting line reaches a length of 20m, a second train should be placed such that the waiting line will decrease faster over time when the two trains are operational. Hence, the guard to node 2 will become active and the system will be described by the dynamics of node 2. The time is then reset and a train is placed, which takes 10 minutes. In those 10 minutes, $\dot{y} = a$ and thus the waiting line length only increases. This is because the Python is non-operational whenever a train is placed/removed and hence no visitors can ride the roller coaster.

When $x(t) \geq 10$, the guard to node 3 becomes active, meaning that the placement/removal of the train is finished. When the Python is operational with two trains, the waiting line length decreases faster than in node 1 because more people get on the roller coaster on average: $\dot{y}(t) = a - b_2$, for $b_2 > b_1$. Finally, when the length of the line gets beneath the 20m again, the process is reversed: a train is removed, which again takes 10 minutes, and the system is described by the dynamics of node 1 again.

Step 1.2

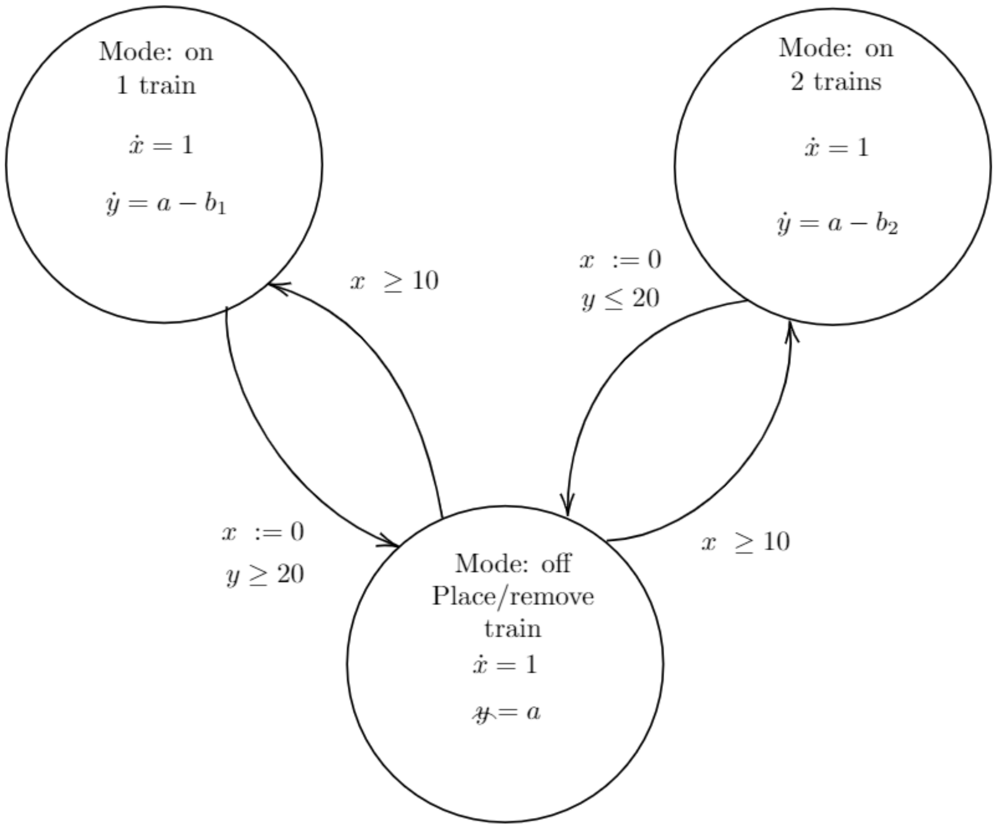

Figure 3: Hybrid automaton for the waiting line for the Python in the Efteling

Part 2: Energy Management of Microgrids

Step 2.1

The discrete-time piecewise affine (PWA) model of the battery is:

$$x_b(k+1) = \begin{cases} x_b(k) - \eta_d T_s u_b(k) & \text{if } s_b = 0 \text{ (discharged)} \\ x_b(k) - \eta_c T_s u_b(k) & \text{if } s_b = 1 \text{ (charged)} \end{cases}$$

$$y_b(k) = u_b(k)$$

Here, x_b is the stored energy in the battery [kWh], u_b the exchanged power [kW], s_b [-] the operational mode (charge / discharge) and η_c, η_d [-] the charging and discharging efficiency respectively. The sampling time of the system is $T_s = 0.20$ [h]. Because the behaviour of the model is described in discrete time (at a certain time instant k), the sampling time is incorporated in the difference equation. In this way, also the units of the equation are correctly used.

The charging/discharging is considered from the grid side. This means that when the battery is charged (discharged), $s_b(k) = 1$ and $u_b(k) \leq 0$ ($s_b(k) = 0$ and $u_b(k) > 0$).

Step 2.2

Looking at the given constraints, and making use of binary variable $s_b(k)$, the following two formulations are constructed:

$$[s_b(k) = 1] \iff [u_b(k) \leq 0]$$
$$[s_b(k) = 0] \iff [u_b(k) > 0]$$

These can be summarized as follows, for ϵ a small tolerance, typically the machine precision:

$$[s_b(k) = 1] \iff [u_b(k) \leq 0] \quad \text{true if and only if:} \quad \begin{cases} u_b(k) \leq \overline{u_b}(1 - s_b(k)) \\ u_b(k) \geq \epsilon + (\underline{u_b} - \epsilon)s_b(k) \end{cases} \tag{1}$$

Then, introducing $z_b(k) = s_b(k)u_b(k)$, the following linear constraints are obtained:

$$z_b(k) \leq \overline{u_b} s_b(k) \tag{2}$$
$$z_b(k) \geq \underline{u_b} s_b(k) \tag{3}$$
$$z_b(k) \leq u_b(k) - \underline{u_b}(1 - s_b(k)) \tag{4}$$
$$z_b(k) \geq u_b(k) - \overline{u_b}(1 - s_b(k)) \tag{5}$$

Looking at the minimum and maximum given in the assignment, another two constraints can be set up:

$$0 \leq x_b(k) \tag{6}$$
$$x_b(k) \leq \overline{x_b} \tag{7}$$

For $u_b(k)$, a maximum and minimimum $\overline{u}_b, \underline{u}_b$ is defined as well. However, these constraints are not taken into account in the same way that the maximum and minimum for $x_b(k)$ is defined in constraints 6, 7, because the constraints for $u_b(k)$ are indirectly already given in the linear constraints 2 to 5.

Rewriting the PWA from step 2.1 to get one equation:

$$x_b(k+1) = -s_b(k)\eta_c T_s u_b(k) + \eta_d T_s u_b(k)(s_b(k) - 1) + x_b(k)$$

The battery can be written as a MLD system given by:

$$x_b(k+1) = A^b x_b(k) + B_1^b u_b(k) + B_2^b \delta_b(k) + B_3^b z_b(k) + B_4^b \tag{8}$$
$$x_b(k+1) = \underbrace{1}_{A^b} x_b(k) \underbrace{-T_s\eta_d}_{B_1^b} u_b(k) + \underbrace{T_s(\eta_d - \eta_c)}_{B_3^b} z(k) \tag{9}$$

Where $A^b = 1, B_1^b = -T_s\eta_d, B_2^b = 0, B_3^b = T_s(\eta_d - \eta_c), B_4^b = 0$.

This equation 9 is subjected to the constraints 2 to 5, which together with constraints 1, constraint 6 and constraint 7 form all the constraints to the MLD problem. Firstly, these constraints are rewritten to equations in the right form. Secondly, the constraints are rewritten to matrix representation for the MLD formulation. The following constraints in correct form are obtained:

(1) $u_b(k) + \overline{u_b}s_b(k) \leq \overline{u_b}$

(2) $(\underline{u_b} - \epsilon)s_b(k) - u_b(k) \leq -\epsilon$

(3) $-x_b(k) \leq 0$

(4) $x_b(k) \leq \overline{x_b}$

(5) $z_b(k) - \overline{u_b}s_b(k) \leq 0$

(6) $\underline{u_b}s_b(k) - z_b(k) \leq 0$

(7) $z_b(k) - u_b(k) - \underline{u_b}s_b(k) \leq -\underline{u_b}$

(8) $u_b(k) - z_b(k) + \overline{u_b}s_b(k) \leq \overline{u_b}$

The following constraint equation and constraint matrices for the battery are obtained:

$$E_1^b x_b(k) + E_2^b u_b(k) + E_3^b s_b(k) + E_4^b z_b(k) \leq g_5^b \tag{10}$$

$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{E_1^b} x_b(k) + \underbrace{\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}}_{E_2^b} u_b(k) + \underbrace{\begin{bmatrix} \overline{u_b} \\ \underline{u_b} - \epsilon \\ 0 \\ 0 \\ -\overline{u_b} \\ \underline{u_b} \\ -\underline{u_b} \\ \overline{u_b} \end{bmatrix}}_{E_3^b} s_b(k) + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}}_{E_4^b} z(k) \leq \underbrace{\begin{bmatrix} \overline{u_b} \\ -\epsilon \\ 0 \\ \overline{x_b} \\ 0 \\ 0 \\ -\underline{u_b} \\ \overline{u_b} \end{bmatrix}}_{g_5^b} \tag{11}$$

For $E_1^b, E_2^b, E_3^b, E_4^b, g_5^b \in \mathbb{R}^{8\times 1}$.

Step 2.3

The fuel consumption of the diesel generator is given by the following function:

$$f\left(u_{\mathrm{d}}(k)\right) = \begin{cases} u_{\mathrm{d}}^2(k) + 4 \\ 4u_{\mathrm{d}}(k) \\ -9.44u_{\mathrm{d}}^3(k) + 166.06u_{\mathrm{d}}^2(k) - 948.22u_{\mathrm{d}}(k) + 1790.28 \\ -11.78u_{\mathrm{d}}(k) + 132.44 \\ 4.01\left(u_{\mathrm{d}}(k) - 10.47\right)^2 + 17.79 \end{cases}$$

With $f(u_d(k))$ the consumed fuel of the diesel generator at time step k in [kg/h] and $u_d(k)$ the output power of the diesel generator at time step k in [kW].

This nonlinear function will be approximated with the PWA function $\hat{f} : [0, \overline{u}_d] \longrightarrow \mathbb{R}$, which is divided in the following four regions:

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

The parameters a_i and b_i for $i \in 1, 2, 3, 4$ in the PWA approximation $\hat{f}$ are determined by minimizing the squared area between f and $\hat{f}$, or equivalently:

$$\int_0^{\overline{u_d}} (f(u_d) - \hat{f}(u_d))^2 du_d$$

To minimize, this integral is divided into the four PWA approximation regions: 0 to $u_1 = 5$, u_1 to $u_2 = 6.5$, u_2 to $u_3 = 11$ and u_3 to 15:

$$\begin{aligned}
&(1) \int_0^2 (u_d^2(k) + 4 - a_1 - b_1 u_d(k))^2 du_d + \int_2^5 (4u_d(k) - a_1 - b_1 u_d(k))^2 du_d \\
&(2) + \int_5^{6.5} (-9.44u_d^3(k) + 166.06u_d^2(k) - 948.22u_d(k) + 1790.28 - a_2 - b_2 u_d(k))^2 du_d \\
&(3) + \int_{6.5}^{7} (-9.44u_d^3(k) + 166.06u_d^2(k) - 948.22u_d(k) + 1790.28 - a_3 - b_3 u_d(k))^2 du_d \\
&+ \int_7^9 (-11.78u_d(k) + 132.44 - a_3 - b_3 u_d(k))^2 du_d + \int_9^{11} (4.01(u_d(k) - 10.47)^2 + 17.79 - a_3 - b_3 u_d(k))^2 du_d \\
&(4) + \int_{11}^{15} (4.01(u_d(k) - 10.47)^2 + 17.79 - a_4 - b_4 u_d(k))^2 du_d
\end{aligned}$$

To minimize this squared area, the integrals are solved analytically. The four regions are minimized separately by first calculating the partial derivatives to a_i and b_i. Secondly, these partial derivatives are set equal to 0 and solved. For example, defining the two integrals in region 1 as a function f_1, $\nabla f_1(a_1, b_1) = 0$ is computed and the minimized values for a_1 and b_1 are obtained. Following this procedure, this results in the following values for a_i, b_i:

	Precise value	Approximate value
a_1	$\frac{136}{75}$	1.813
a_2	$\frac{-22732801477233947}{247390116249600}$	-91.891
a_3	$\frac{3591809146630509439}{32061759065948160}$	112.028
a_4	$\frac{-121019335510265749}{562949953421312}$	-21.4974

	Precise value	Approximate value
b_1	$\frac{436}{125}$	3.488
b_2	$\frac{1357800653035933}{61847529062400}$	21.954
b_3	$\frac{-1472321153042095969}{160308795329740800}$	-9.184
b_4	$\frac{11422592324890509}{562949953421312}$	20.291

To compare the approximated function with the real function for the fuel consumption, the PWA approximation with values found for a_i, b_i is plotted against the real function, see figure 4.

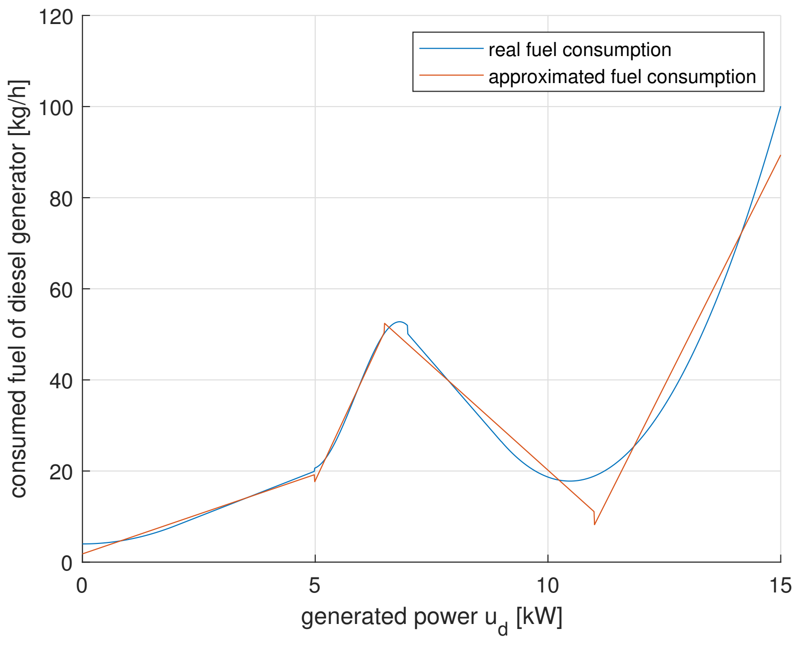

Figure 4: Comparison of real with approximated fuel consumption [kg/h], for optimal values of a_i, b_i for $i = 1, 2, 3, 4$

Step 2.4

The PWA approximation derived in step 2.3 is now recalculated, only now with the bounds u_i of the four PWA approximation regions not fixed but variable. Now, the objective is to minimize:

$$\int_0^{\overline{u}_\mathrm{d}} \left(f\left(u_\mathrm{d}\right) - \hat{f}\left(u_\mathrm{d}\right)\right)^2 du_\mathrm{d}$$

The function should be minimized over 11 variables: a_i, b_i, u_1, u_2, u_3 for $i = 1, 2, 3, 4$. Both functions, $f(u_d(k))$ and $\hat{f}(u_d(k))$, are first discretized on the interval $u_d(k) \in [0 \quad 15]$ with step size 0.001. The values that should be estimated are stored in vector x and the objective function to minimize is created. The integral from the objective is calculated using trapezoidal numerical integration. Using the algorithm `fmincon` in MATLAB, the optimal vector x is found. The algorithm finds the minimum of a constrained nonlinear multivariable function, where the constraints are given by:

(1) $u_3 \leq \overline{u}_d$

(2) $u_1 - u_2 \leq 0$

(3) $u_2 - u_3 \leq 0$

(4) $-u_1 \leq 0$

These can be summarized as $Ax \leq b$, where the matrices and vector are defined as:

$$x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} \overline{u}_d \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

To make sure the algorithm does not get stuck in a local minimum, 21 different initial vectors x_0 are defined as a starting point, with values close to the estimated values from step 2.3. The minimum value found for the objective function is 118.5073 [kg/h], for the following values for the variables:

	Approximate value
a_1	1.8108
a_2	-86.7707
a_3	102.9772
a_4	-229.6789

	Approximate value
b_1	3.4880
b_2	20.9957
b_3	-8.1265
b_4	21.3577

	Approximate value
u_1	4.9860
u_2	6.8700
u_3	11.3019

To compare the approximated function with the real function for the fuel consumption, the PWA approximation with values found for a_i, b_i, u_1, u_2, u_3 is plotted against the real function, see figure 5.

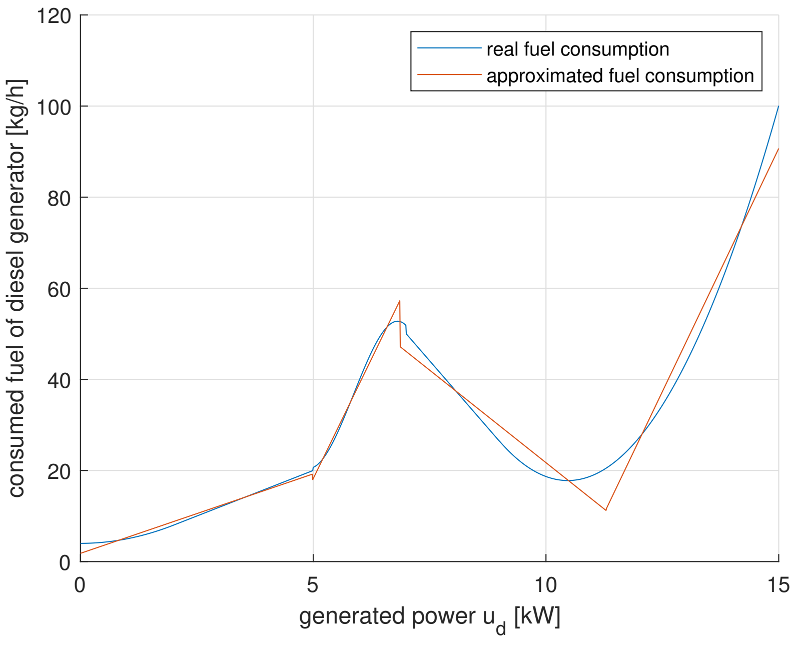

Figure 5: Comparison of real with approximated fuel consumption [kg/h], for optimal values of a_i, b_i, u_1, u_2, u_3 for $i = 1, 2, 3, 4$

Finally, the Root Mean Square Error (RMSE) between the real and approximated fuel consumption function is computed, which equals 2.8145 [kg/h]. The RMSE is calculated as well for step 2.3, where the approximation was done without optimizing u_1, u_2, u_3. The RMSE for step 2.3 equals 2.9046 kg/h. Thus, in step 2.4, a 3.10% lower error than in step 2.3 is obtained. Therefore, it can be concluded that the values obtained in this step are more optimal.

Step 2.5

The PWA model for the diesel generator is given below, with $\hat{f}(u_d(k))$ the PWA approximation of the consumed fuel of the diesel generator as calculated in step 2.3:

$$x_d(k+1) = \begin{cases} x_d(k) + R_f T_s - \hat{f}(u_d(k))T_s & \text{if } s_d(k) = 1 \\ x_d(k) + R_f T_s & \text{if } s_d(k) = 0 \end{cases}$$

Filling in $\hat{f}(u_d(k))$, the PWA model for the diesel generator becomes:

$$x_d(k+1) = \begin{cases} x_d(k) + R_f T_s - a_1 T_s - b_1 T_s u_d(k) & \text{if } s_d(k) = 1, \quad 0 \leq u_d(k) \leq u_1 \\ x_d(k) + R_f T_s - a_2 T_s - b_2 T_s u_d(k) & \text{if } s_d(k) = 1, \quad u_1 \leq u_d(k) \leq u_2 \\ x_d(k) + R_f T_s - a_3 T_s - b_3 T_s u_d(k) & \text{if } s_d(k) = 1, \quad u_2 \leq u_d(k) \leq u_3 \\ x_d(k) + R_f T_s - a_4 T_s - b_4 T_s u_d(k) & \text{if } s_d(k) = 1, \quad u_3 \leq u_d(k) \leq \overline{u}_d \\ x_d(k) + R_f T_s & \text{if } s_d = 0 \end{cases} \tag{13}$$

Here, $x_d(k)$ is the fuel level in the tank, or the remaining fuel of the generator [kg], $u_d(k)$ the generated power by the generator [kW], $s_d(k)$ [-] the operational mode (on / off) of the generator and R_f [kg/h] the filling rate of the fuel tank. R_f can be considered a constant and the value of $\overline{u}_d = 15$.

Step 2.6

The objective is to construct a MLD model of the diesel generator.

Constraints
First, a binary variable $s_d(k)$ is introduced, which indicates whether the generator is switched on or off:

$$[s_d(k) = 1] \iff [u_d(k) > 0]$$
$$[s_d(k) = 0] \iff [u_d(k) = 0]$$

For the battery, it holds that $s_b(k) = \delta_b(k)$ and $z(k) = s_b(k)u_b(k)$. However, for the diesel generator, four new binary variables are introduced, such that the $\delta_d(k)$ and $z_d(k)$ vectors are defined as follows:

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

For these new binary variables, the following is desired:

$$\begin{array}{lll} [\delta_1 = 1] \text{ if } [0 \leq u_d(k) < u_1] & \text{else} & [\delta_1 = 0] \\ [\delta_2 = 1] \text{ if } [u_1 \leq u_d(k) < u_2] & \text{else} & [\delta_2 = 0] \\ [\delta_3 = 1] \text{ if } [u_2 \leq u_d(k) < u_3] & \text{else} & [\delta_3 = 0] \\ [\delta_4 = 1] \text{ if } [u_2 \leq u_d(k) < 15] & \text{else} & [\delta_4 = 0] \end{array}$$

This gives rise to equality constraint:

$$\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) = s_d(k)$$

Which can be rewritten to the first inequality constraint:

$$\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) \leq 1 \tag{14}$$

Update equation
With the newly defined vectors $\delta_d(k), z_d(k)$, the PWA from step 2.5, equation 13, should be rewritten to obtain one update equation:

$$\begin{aligned} x_d(k+1) =& x_d(k) + T_s R_f - T_s a_1 \delta_1(k) - T_s a_2 \delta_2(k) - T_s a_3 \delta_3(k) - T_s a_4 \delta_4(k) - T_s b_1 \delta_1(k) u_d(k) - T_s b_2 \delta_2(k) u_d(k) \\ & - T_s b_3 \delta_3(k) u_d(k) - T_s b_4 \delta_4(k) u_d(k) \\ =& x_d(k) + T_s R_f - T_s a_1 \delta_1(k) - T_s a_2 \delta_2(k) - T_s a_3 \delta_3(k) - T_s a_4 \delta_4(k) - T_s b_1 z_1(k) - T_s b_2 z_2(k) \\ & - T_s b_3 z_3(k) - T_s b_4 z_4(k) \end{aligned} \tag{15}$$

This equation can be summarized as follows:

$$x_d(k+1) = A^d x_d(k) + B_1^d u_d(k) + B_2^d \delta_d(k) + B_3^d z_d(k) + B_4^d \tag{16}$$

Here, the matrices are defined as:

$$\begin{aligned} &A^d = 1 \qquad B_1^d = 0 \qquad B_2^d = \begin{bmatrix} -T_s a_1 & -T_s a_2 & -T_s a_3 & -T_s a_4 \end{bmatrix} \\ &B_3^d = \begin{bmatrix} -T_s b_1 & -T_s b_2 & -T_s b_3 & -T_s b_4 \end{bmatrix} \qquad B_4^d = T_s R_f \end{aligned} \tag{17}$$

For the following vectors:

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

Constraints
Looking at the minimum and maximum given in the assignment, another two constraints can be set up:

$$\underline{x}_d \leq x_d(k) \implies -x_d(k) \leq -\underline{x}_d \tag{18}$$
$$x_d(k) \leq \overline{x}_d \tag{19}$$
$$\underline{u}_d \leq u_d(k) \implies -u_d(k) \leq -\underline{u}_d \tag{20}$$
$$u_d(k) \leq \overline{u}_d \tag{21}$$

The linear inequalities that rise from writing the nonlinear system as a linear system are written for each $\delta_i(k), z_i(k)$ separately:

$$z_1(k) \leq u_1\delta_1(k) \implies -u_1\delta_1(k) + z_1(k) \leq 0 \tag{22}$$
$$z_1(k) \geq 0 \cdot \delta_1(k) \implies -z_1(k) \leq 0 \tag{23}$$
$$z_1(k) \leq u_d(k) - 0 \cdot (1 - \delta_1(k)) \implies -u_d(k) + z_1(k) \leq 0 \tag{24}$$
$$z_1(k) \geq u_d(k) - u_1(1 - \delta_1(k)) \implies u_d(k) + u_1\delta_1(k) - z_1(k) \leq u_1 \tag{25}$$

$$z_2(k) \leq u_2\delta_2(k) \implies -u_2\delta_2(k) + z_2(k) \leq 0 \tag{26}$$
$$z_2(k) \geq u_1\delta_2(k) \implies u_1\delta_2(k) - z_2(k) \leq 0 \tag{27}$$
$$z_2(k) \leq u_d(k) - u_1(1 - \delta_2(k)) \implies -u_d(k) - u_1\delta_2(k) + z_2(k) \leq 0 \tag{28}$$
$$z_2(k) \geq u_d(k) - u_2(1 - \delta_2(k)) \implies u_d(k) + u_2\delta_2(k) - z_2(k) \leq u_2 \tag{29}$$

$$z_3(k) \leq u_3\delta_3(k) \implies -u_3\delta_3(k) + z_3(k) \leq 0 \tag{30}$$
$$z_3(k) \geq u_2\delta_3(k) \implies u_2\delta_3(k) - z_3(k) \leq 0 \tag{31}$$
$$z_3(k) \leq u_d(k) - u_2(1 - \delta_3(k)) \implies -u_d(k) - u_2\delta_3(k) + z_3(k) \leq -u_2 \tag{32}$$
$$z_3(k) \geq u_d(k) - u_3(1 - \delta_3(k)) \implies u_d(k) + u_3\delta_3(k) - z_3(k) \leq u_3 \tag{33}$$

$$z_4(k) \leq \overline{u}_d\delta_4(k) \implies -\overline{u}_d\delta_4(k) + z_4(k) \leq 0 \tag{34}$$
$$z_4(k) \geq u_3\delta_4(k) \implies u_3\delta_4(k) - z_4(k) \leq 0 \tag{35}$$
$$z_4(k) \leq u_d(k) - u_3(1 - \delta_4(k)) \implies -u_d(k) - u_3\delta_4(k) + z_4(k) \leq -u_3 \tag{36}$$
$$z_4(k) \geq u_d(k) - \overline{u}_d(1 - \delta_4(k)) \implies u_d(k) + \overline{u}_d\delta_4(k) - z_4(k) \leq \overline{u}_d \tag{37}$$
$$\tag{38}$$

Next, the binary variables are related to the fuel consumption, for ϵ a small tolerance, typically the machine precision. This relation, for the first binary variable only, is as follows:

$$[\delta_1 = 1] \iff [u_d(k) \geq 0]$$

true if and only if

$$u_d(k) \geq \epsilon\delta_1(k) \implies -u_d(k) + \epsilon\delta_1(k) \leq 0 \tag{39}$$
$$u_d(k) - (\overline{u}_d - \underline{u}_d + \epsilon)\delta_1(k) \leq u_1 - \epsilon \implies u_d(k) - (\overline{u}_d - \underline{u}_d + \epsilon) \leq \underline{u}_d - \epsilon \tag{40}$$
$$\tag{41}$$

$$[\delta_2 = 1] \iff [u_d(k) \geq u_1]$$
true if and only if
$$u_d(k) \geq u_1\delta_2(k) \implies -u_d(k) + u_1\delta_2(k) \leq 0 \tag{42}$$
$$-(\overline{u}_d - u_1 + \epsilon)\delta_2(k) \leq -(u_d - u_1) - \epsilon \implies u_d(k) - (\overline{u}_d - u_1 + \epsilon)\delta_2(k) \leq u_1 - \epsilon \tag{43}$$
$$\tag{44}$$

$$[\delta_3 = 1] \iff [u_d(k) \geq u_2]$$
true if and only if
$$u_d(k) \geq u_2\delta_3(k) \implies -u_d(k) + u_2\delta_3(k) \leq 0 \tag{45}$$
$$-(\overline{u}_d - u_2 + \epsilon)\delta_3(k) \leq -(u_d(k) - u_2) - \epsilon \implies u_d(k) - (\overline{u}_d - u_2 + \epsilon)\delta_3(k) \leq u_2 - \epsilon \tag{46}$$
$$\tag{47}$$

$$[\delta_4 = 1] \iff [u_d(k) \geq u_3]$$
true if and only if
$$u_d(k) \geq u_3\delta_4(k) \implies -u_d(k) + u_3\delta_4(k) \leq 0 \tag{48}$$
$$-(\overline{u}_d - u_3 + \epsilon)\delta_4(k) \leq -(u_d(k) - u_3) - \epsilon \implies u_d(k) - (\overline{u}_d - u_3 + \epsilon)\delta_4(k) \leq u_3 - \epsilon \tag{49}$$
$$\tag{50}$$

The final constraints that arise in the MLD model of the diesel generator are given by constraints 14, 18, 21, linear constraints 22 to 37 and binary constraints 39 to 49:

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

(2) $-x_d(k) \leq -\underline{x}_d$

(3) $x_d(k) \leq \overline{x}_d$

(4) $-u_1\delta_1(k) + z_1(k) \leq 0$

(5) $-z_1(k) \leq 0$

(6) $-u_d(k) + z_1(k) \leq 0$

(7) $u_d(k) + u_1\delta_1(k) - z_1(k) \leq u_1$

(8) $-u_d(k) + \epsilon\delta_1(k) \leq 0$

(9) $u_d(k) - (\overline{u}_d - \underline{u}_d + \epsilon) \leq \underline{u}_d - \epsilon$

(10) $-u_2\delta_2(k) + z_2(k) \leq 0$

(11) $u_1\delta_2(k) - z_2(k) \leq 0$

(12) $-u_d(k) - u_1\delta_2(k) + z_2(k) \leq -u_1$

(13) $u_d(k) + u_2\delta_2(k) - z_2(k) \leq u_2$

(14) $-u_d(k) + u_1\delta_2(k) \leq 0$

(15) $u_d(k) - (\overline{u}_d - u_1 + \epsilon)\delta_2(k) \leq u_1 - \epsilon$

(16) $-u_3\delta_3(k) + z_3(k) \leq 0$

(17) $u_2\delta_3(k) - z_3(k) \leq 0$

(18) $-u_d(k) - u_2\delta_3(k) + z_3(k) \leq -u_2$

(19) $u_d(k) + u_3\delta_3(k) - z_3(k) \leq u_3$

(20) $-u_d(k) + u_2\delta_3(k) \leq 0$

(21) $u_d(k) - (\overline{u}_d - u_2 + \epsilon)\delta_3(k) \leq u_2 - \epsilon$

(22) $-\overline{u}_d\delta_4(k) + z_4(k) \leq 0$

(23) $u_3\delta_4(k) - z_4(k) \leq 0$

(24) $-u_d(k) - u_3\delta_4(k) + z_4(k) \leq -u_3$

(25) $u_d(k) + \overline{u}_d\delta_4(k) - z_4(k) \leq \overline{u}_d$

(26) $-u_d(k) + u_3\delta_4(k) \leq 0$

(27) $u_d(k) - (\overline{u}_d - u_3 + \epsilon)\delta_4(k) \leq u_3 - \epsilon$

These constraints can be summarized in the following equation:

$$E_1^d x_d(k) + E_2^d u_d(k) + E_3^d \delta_d(k) + E_4^d z_d(k) \leq g_5^d \tag{51}$$

Here, for $E_1^d, E_2^d, g_5^d \in \mathbb{R}^{27\times 1}$ and $E_3^d, E_4^d \in \mathbb{R}^{27\times 4}$, the matrices are defined as:

$$E_1^d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad E_2^d = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad E_3^d = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -u_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ u_1 & 0 & 0 & 0 \\ \epsilon & 0 & 0 & 0 \\ -(\overline{u}_d - \underline{u}_d + \epsilon) & 0 & 0 & 0 \\ 0 & -u_2 & 0 & 0 \\ 0 & u_1 & 0 & 0 \\ 0 & -u_1 & 0 & 0 \\ 0 & u_2 & 0 & 0 \\ 0 & u_1 & 0 & 0 \\ 0 & -(\overline{u}_d - u_1 + \epsilon) & 0 & 0 \\ 0 & 0 & -u_3 & 0 \\ 0 & 0 & u_2 & 0 \\ 0 & 0 & -u_2 & 0 \\ 0 & 0 & u_3 & 0 \\ 0 & 0 & u_2 & 0 \\ 0 & 0 & -(\overline{u}_d - u_2 + \epsilon) & 0 \\ 0 & 0 & 0 & -\overline{u}_d \\ 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & -u_3 \\ 0 & 0 & 0 & \overline{u}_d \\ 0 & 0 & 0 & u_3 \\ 0 & 0 & 0 & -(\overline{u}_d - u_3 + \epsilon) \end{bmatrix} \quad E_4^d = \begin{bmatrix} 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 & 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 & 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 & 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 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad g_5^d = \begin{bmatrix} \overline{u}_d \\ \epsilon \\ 1 \\ -\underline{x}_d \\ \overline{x}_d \\ 0 \\ 0 \\ 0 \\ u_1 \\ 0 \\ \underline{u}_d - \epsilon \\ 0 \\ 0 \\ -u_1 \\ u_2 \\ 0 \\ u_1 - \epsilon \\ 0 \\ 0 \\ -u_2 \\ u_3 \\ 0 \\ u_2 - \epsilon \\ 0 \\ 0 \\ -u_3 \\ \overline{u}_d \\ 0 \\ u_3 - \epsilon \end{bmatrix} \tag{52}$$

Step 2.7

See MATLAB file in the Appendix.

Step 2.8

MPC Problem

The goal is to recast the MPC optimization problem as a mixed-integer linear programming (MILP) problem. The MPC problem is given by the following three update and constraint equations.

Update equations

$$x_{b,1}(k+1) = A^{b,1}x_{b,1}(k) + B_1^{b,1}u_{b,1}(k) + B_3^{b,1}z_{b,1}(k) \tag{53}$$
$$x_{b,2}(k+1) = A^{b,2}x_{b,2}(k) + B_1^{b,2}u_{b,2}(k) + B_3^{b,2}z_{b,2}(k) \tag{54}$$
$$x_d(k+1) = A^d x_d(k) + B_1^d u_d(k) + B_2^d \delta_d(k) + B_3^d z_d(k) + B_4^d \tag{55}$$

For:

$$z_{b,1}(k) = s_{b,1}(k)u_{b,1}(k) \qquad z_{b,2}(k) = s_{b,2}(k)u_{b,2}(k) \qquad \delta_d(k) = \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} \qquad z_d(k) = \begin{bmatrix} \delta_1(k)u_d(k) \\ \delta_2(k)u_d(k) \\ \delta_3(k)u_d(k) \\ \delta_4(k)u_d(k) \end{bmatrix} \tag{56}$$

Constraint equations

$$E_1^{b,1}x_{b,1}(k) + E_2^{b,1}u_{b,1}(k) + E_3^{b,1}\delta_{b,1}(k) + E_4^{b,1}z_{b,1}(k) \leq g_5^{b,1} \tag{57}$$
$$E_1^{b,2}x_{b,2}(k) + E_2^{b,2}u_{b,2}(k) + E_3^{b,2}\delta_{b,2}(k) + E_4^{b,2}z_{b,2}(k) \leq g_5^{b,2} \tag{58}$$
$$E_1^d x_d(k) + E_2^d u_d(k) + E_3^d \delta_d(k) + E_4^d z_d(k) \leq g_5^d \tag{59}$$

For:

$$\delta_{b,1}(k) = s_{b,1}(k) \qquad \delta_{b,2}(k) = s_{b,2}(k)$$

Writing MPC problem in explicit form

The update and inequality equations should be written in explicit form. This is done for the batteries and diesel generator separately. Afterwards, both the update and inequality equations can be taken together to get two expressions.

Batteries
First, the following vectors are defined, where i denotes either 1 or 2, referring to the first and second battery respectively. Furthermore, N_p denotes the control horizon. All vectors have dimension $\mathbb{R}^{N_p \times 1}$:

$$\tilde{\delta}_{b,i}(k) = \begin{bmatrix} \hat{\delta}_{b,i}(k|k) \\ \vdots \\ \hat{\delta}_{b,i}(k+N_p-1|k) \end{bmatrix} \qquad \tilde{u}_{b,i}(k) = \begin{bmatrix} u_{b,i}(k) \\ \vdots \\ u_{b,i}(k+N_p-1) \end{bmatrix} \tilde{z}_{b,i}(k) = \begin{bmatrix} \hat{z}_{b,i}(k|k) \\ \vdots \\ \hat{z}_{b,i}(k+N_p-1|k) \end{bmatrix}$$

$$\tilde{x}_{b,i}(k) = \begin{bmatrix} \hat{x}_{b,i}(k+1|k) \\ \vdots \\ \hat{x}_{b,i}(k+N_p|k) \end{bmatrix}$$

Second, all components of vector $\tilde{x}_{b,i}(k)$, $\tilde{\delta}_{b,i}(k)$ and $\tilde{z}_{b,i}(k)$ are expressed as a function of $x(k)$ and of all components in vector $u(k)$. To achieve this, the inequality equations 57, 58 are rewritten to determine $\hat{\delta}(k|k), \hat{z}(k|k)$:

$$E_1^{b,1}x_{b,1}(k) + E_2^{b,1}u_{b,1}(k) + E_3^{b,1}\hat{\delta}_{b,1}(k|k) + E_4^{b,1}\hat{z}_{b,1}(k|k) \leq g_5^{b,1} \tag{60}$$
$$E_1^{b,2}x_{b,2}(k) + E_2^{b,2}u_{b,2}(k) + E_3^{b,2}\hat{\delta}_{b,2}(k|k) + E_4^{b,2}\hat{z}_{b,2}(k|k) \leq g_5^{b,2} \tag{61}$$

Then, update equations 53, 54 are used to determine components of $\hat{x}_{b,i}(k+1|k)$:

$$\hat{x}_{b,1}(k+1|k) = A^{b,1}x_{b,1}(k) + B_1^{b,1}u_{b,1}(k) + B_3^{b,1}\hat{z}_{b,1}(k|k) \tag{62}$$
$$\hat{x}_{b,2}(k+1|k) = A^{b,2}x_{b,2}(k) + B_1^{b,2}u_{b,2}(k) + B_3^{b,2}\hat{z}_{b,2}(k|k) \tag{63}$$

Again, inequality equations 57, 58 are used to determine $\hat{\delta}_{b,i}(k+1|k)$, by looking at time step k+1:

$$E_1^{b,1}\hat{x}_{b,1}(k+1|k) + E_2^{b,1}u_{b,1}(k+1) + E_3^{b,1}\hat{\delta}_{b,1}(k+1|k) + E_4^{b,1}\hat{z}_{b,1}(k+1|k) \leq g_5^{b,1}$$
$$E_1^{b,2}\hat{x}_{b,2}(k+1|k) + E_2^{b,2}u_{b,2}(k+1) + E_3^{b,2}\hat{\delta}_{b,2}(k+1|k) + E_4^{b,2}\hat{z}_{b,2}(k+1|k) \leq g_5^{b,2}$$

Here, $\hat{x}_{b,i}(k+1|k)$ can be eliminated using equations 62 and 63:

$$E_1^{b,1}A^{b,1}x_{b,1}(k) + E_1^{b,1}B_1^{b,1}u_{b,1}(k) + E_2^{b,1}u_{b,1}(k+1) + E_1^{b,1}B_3^{b,1}\hat{z}_{b,1}(k|k)) + E_4^{b,1}\hat{z}_{b,1}(k+1|k) + E_3^{b,1}\hat{\delta}_{b,1}(k+1|k) \leq g_5^{b,1} \tag{64}$$
$$E_1^{b,2}A^{b,2}x_{b,2}(k) + E_1^{b,2}B_1^{b,2}u_{b,2}(k) + E_2^{b,2}u_{b,2}(k+1) + E_1^{b,2}B_3^{b,2}\hat{z}_{b,2}(k|k)) + E_4^{b,2}\hat{z}_{b,2}(k+1|k) + E_3^{b,2}\hat{\delta}_{b,2}(k+1|k) \leq g_5^{b,2} \tag{65}$$

Then, we use update equations 53 and 54 again to determine the next time step $\hat{x}_{b,i}(k+2|k)$:

$$\hat{x}_{b,1}(k+2|k) = A^{b,1}\hat{x}_{b,1}(k+1|k) + B_1^{b,1}u_{b,1}(k+1) + B_3^{b,1}\hat{z}_{b,1}(k+1|k)$$
$$\hat{x}_{b,2}(k+2|k) = A^{b,2}\hat{x}_{b,2}(k+1|k) + B_1^{b,2}u_{b,2}(k+1) + B_3^{b,2}\hat{z}_{b,2}(k+1|k)$$

Here, $\hat{x}_{b,i}(k+1|k)$ can be eliminated using equations 62 and 63:

$$\hat{x}_{b,1}(k+2|k) = (A^{b,1})^2x_{b,1}(k) + A^{b,1}B_1^{b,1}u_{b,1}(k) + B_1^{b,1}u_{b,1}(k+1) + A^{b,1}B_3^{b,1}\hat{z}_{b,1}(k|k) + B_3^{b,1}\hat{z}_{b,1}(k+1|k) \tag{66}$$
$$\hat{x}_{b,2}(k+2|k) = (A^{b,2})^2x_{b,2}(k) + A^{b,2}B_1^{b,2}u_{b,2}(k) + B_1^{b,2}u_{b,2}(k+1) + A^{b,2}B_3^{b,2}\hat{z}_{b,2}(k|k) + B_3^{b,2}\hat{z}_{b,2}(k+1|k) \tag{67}$$

When continuing this way, the update and inequality equations for $\hat{x}_{b,i}(k+3|k)$ can be determined and when continuing even further, the update and inequality equations for $\hat{x}_{b,i}(k+N_p|k)$. The general form of the equations is the following:

$$\begin{aligned} &E_1^{b,i}(A^{b,i})^\ell x_{b,i}(k) + E_1^{b,i}(A^{b,i})^{\ell-1}B_1^{b,i}u_{b,i}(k) + E_1^{b,i}(A^{b,i})^{\ell-2}B_1^{b,i}u_{b,i}(k+1) + \ldots + E_1^{b,i}B_1^{b,i}u_{b,i}(k+\ell-1) \\ &\quad + E_2^{b,i}u_{b,i}(k+\ell) + E_1^{b,i}(A^{b,i})^{\ell-1}B_3^{b,i}\hat{z}_{b,i}(k|k) + E_1^{b,i}(A^{b,i})^{\ell-2}B_3^{b,i}\hat{z}_{b,i}(k+1|k) + \ldots \\ &\quad + E_1^{b,i}B_3^{b,i}\hat{z}_{b,i}(k+\ell-1) + E_4^{b,i}\hat{z}_{b,i}(k+\ell|k) + E_3^{b,i}\hat{\delta}_{b,i}(k+\ell|k) \leqslant g_5^{b,i} \end{aligned} \tag{68}$$

$$\begin{aligned} \hat{x}_{b,i}(k+\ell+1|k) =& (A^{b,i})^{\ell+1}x_{b,i}(k) + (A^{b,i})^\ell B_1^{b,i}u_{b,i}(k) + (A^{b,i})^{\ell-1}B_1^{b,i}u_{b,i}(k+1) + \ldots + A^{b,i}B_1^{b,i}u_{b,i}(k+\ell-1) \\ &+ B_1^{b,i}u_{b,i}(k+\ell) + (A^{b,i})^\ell B_3^{b,i}\hat{z}_{b,i}(k|k) + (A^{b,i})^{\ell-1}B_3^{b,i}\hat{z}_{b,i}(k+1|k) + \ldots \\ &+ A^{b,i}B_3^{b,i}\hat{z}_{b,i}(k+\ell-1|k) + B_3^{b,i}\hat{z}_{b,i}(k+\ell|k) \end{aligned} \tag{69}$$

for $\ell = 0, \ldots, N_\mathrm{p}-1$.

Diesel generator
For the diesel generator, first the following vectors are defined. Vectors $\tilde{u}_d(k), \tilde{x}_d(k)$ have dimension $\mathbb{R}^{N_p\times 1}$, while vectors $\tilde{\delta}_d(k), \tilde{z}_d(k)$ have dimension $\mathbb{R}^{4N_p\times 1}$ (as can be seen in equation 56, each δ_d, z_d has length 4): :

$$\tilde{\delta}_d(k) = \begin{bmatrix} \hat{\delta}_d(k|k) \\ \vdots \\ \hat{\delta}_d(k+N_p-1|k) \end{bmatrix} \qquad \tilde{u}_d(k) = \begin{bmatrix} u_{b,i}(k) \\ \vdots \\ u_d(k+N_p-1) \end{bmatrix} \tilde{z}_d(k) = \begin{bmatrix} \hat{z}_d(k|k) \\ \vdots \\ \hat{z}_d(k+N_p-1|k) \end{bmatrix}$$

$$\tilde{x}_d(k) = \begin{bmatrix} \hat{x}_d(k+1|k) \\ \vdots \\ \hat{x}_d(k+N_p|k) \end{bmatrix}$$

Next, the derivation of the update and inequality equations for each time step was executed in the same way as for the batteries. The general form of the expressions which is then obtained, is the following:

$$\begin{aligned}
&E_1^d(A^d)^\ell x_d(k) + E_1^d(A^d)^{\ell-1}B_1^d u_d(k) + E_1^d(A^d)^{\ell-2}B_1^d u_d(k+1) + \ldots + E_1^d B_1^d u_d(k+\ell-1) + E_2^d u_d(k+\ell)\\
&\quad + E_1^d(A^d)^{\ell-1}B_3^d \hat{z}_d(k|k) + E_1^d(A^d)^{\ell-2}B_3^d \hat{z}_d(k+1|k) + \ldots + E_1^d B_3^d \hat{z}_d(k+\ell-1) + E_4^d \hat{z}_d(k+\ell|k)\\
&\quad + E_1^d(A^d)^{\ell-1}B_2^d \hat{\delta}_d(k|k) + E_1^d(A^d)^{\ell-2}B_2^d \hat{\delta}_d(k+1|k) + \ldots + E_1^d B_2^d \hat{\delta}_d(k+\ell-1) + E_3^d \hat{\delta}_d(k+\ell|k)\\
&\quad + E_1^d \sum_{n=0}^{\ell}(A^d)^{n-1}B_4^d \leqslant g_5^d
\end{aligned} \tag{70}$$

$$\begin{aligned}
\hat{x}_d(k+\ell+1|k) =& (A^d)^{\ell+1}x_d(k) + (A^d)^\ell B_1^d u_d(k) + (A^d)^{\ell-1}B_1^d u_d(k+1) + \ldots + A^d B_1^d u_d(k+\ell-1)\\
&+ B_1^d u_d(k+\ell) + (A^d)^\ell B_3^d \hat{z}_d(k|k) + (A^d)^{\ell-1}B_3^d \hat{z}_d(k+1|k) + \ldots + A^d B_3^d \hat{z}_d(k+\ell-1|k)\\
&+ B_3^d \hat{z}_d(k+\ell|k) + (A^d)^\ell B_2^d \hat{\delta}_d(k|k) + (A^d)^{\ell-1}B_2^d \hat{\delta}_d(k+1|k) + \ldots + A^d B_2^d \hat{\delta}_d(k+\ell-1|k)\\
&+ B_2^d \hat{\delta}_d(k+\ell|k) + \sum_{n=0}^{\ell}(A^d)^n B_4^d
\end{aligned} \tag{71}$$

for $\ell = 0, \ldots, N_\mathrm{p}-1$. Inequality expression 70 does not hold for $\ell = 0$ and should only be used for all expressions $\ell > 1$.

Batteries and diesel generator together
The estimate vector $\tilde{V}(k)$ is defined as follows:

$$\tilde{V}(k) = \begin{bmatrix} \tilde{V}_{b,1}(k) \\ \tilde{V}_{b,2}(k) \\ \tilde{V}_d(k) \end{bmatrix} \tag{72}$$

Where the submatrices are given by:

$$\tilde{V}_{b,1}(k) = \begin{bmatrix} \hat{\delta}_{b,1}(k|k) \\ \hat{\delta}_{b,1}(k+1|k) \\ \vdots \\ \hat{\delta}_{b,1}(k+N_p-1|k) \\ u_{b,1}(k|k) \\ u_{b,1}(k+1|k) \\ \vdots \\ u_{b,1}(k+N_p-1|k) \\ \hat{z}_{b,1}(k|k) \\ \hat{z}_{b,1}(k+1|k) \\ \vdots \\ \hat{z}_{b,1}(k+N_p-1|k) \end{bmatrix} \quad \tilde{V}_{b,2}(k) = \begin{bmatrix} \hat{\delta}_{b,2}(k|k) \\ \hat{\delta}_{b,2}(k+1|k) \\ \vdots \\ \hat{\delta}_{b,2}(k+N_p-1|k) \\ u_{b,2}(k|k) \\ u_{b,2}(k+1|k) \\ \vdots \\ u_{b,2}(k+N_p-1|k) \\ \hat{z}_{b,2}(k|k) \\ \hat{z}_{b,2}(k+1|k) \\ \vdots \\ \hat{z}_{b,2}(k+N_p-1|k) \end{bmatrix} \quad \tilde{V}_d(k) = \begin{bmatrix} \hat{\delta}_d(k|k) \\ \hat{\delta}_d(k+1|k) \\ \vdots \\ \hat{\delta}_d(k+N_p-1|k) \\ u_d(k|k) \\ u_d(k+1|k) \\ \vdots \\ u_d(k+N_p-1|k) \\ \hat{z}_d(k|k) \\ \hat{z}_d(k+1|k) \\ \vdots \\ \hat{z}_d(k+N_p-1|k) \end{bmatrix} \tag{73}$$

Here, $\tilde{V}_{b,1}(k), \tilde{V}_{b,2}(k) \in \mathbb{R}^{3N_p \times 1}$ and $\tilde{V}_d(k) \in \mathbb{R}^{9N_p \times 1}$. Furthermore, $x(k), \tilde{x}(k)$ are described by:

$$x(k) = \begin{bmatrix} x_{b,1}(k) \\ x_{b,2}(k) \\ x_d(k) \end{bmatrix} \qquad \tilde{x}(k) = \begin{bmatrix} \hat{x}_{b,1}(k+1|k) \\ \vdots \\ \hat{x}_{b,1}(k+N_p|k) \\ \hat{x}_{b,2}(k+1|k) \\ \vdots \\ \hat{x}_{b,2}(k+N_p|k) \\ \hat{x}_d(k+1|k) \\ \vdots \\ \hat{x}_d(k+N_p|k) \end{bmatrix} \tag{74}$$

For $x(k) \in \mathbb{R}^{3\times 1}$ and $\tilde{x}(k) \in \mathbb{R}^{3N_p\times 1}$. With $\tilde{V}(k), x(k), \tilde{x}(k)$ and the equations and inequalities obtained in the previous part, the expressions can be written in a more compact form. This compact form, including the notation of the matrices, is given in the following two subsections: 'Update equations MILP' and 'Constraint equations MILP'.

Update equations MILP

The update equations, given by 69 and 71 for the batteries and diesel generator respectively, can be summarized as follows:

$$\tilde{x}(k) = M_1\tilde{V}(k) + M_2x(k) + M_3 \tag{75}$$

Here, the following holds for the matrices:

$$M_1 = \begin{bmatrix} M_1^{b,1} & 0 & 0 \\ 0 & M_1^{b,2} & 0 \\ 0 & 0 & M_1^d \end{bmatrix} \qquad M_2 = \begin{bmatrix} M_2^{b,1} & 0 & 0 \\ 0 & M_2^{b,2} & 0 \\ 0 & 0 & M_2^d \end{bmatrix} \qquad M_3 = \begin{bmatrix} M_3^{b,1} \\ M_3^{b,2} \\ M_3^d \end{bmatrix}$$

$$M_1^d = \begin{bmatrix} M_1^d(\hat{\delta}_d) & M_1^d(u_d) & M_1^d(\hat{z}_d) \end{bmatrix} \tag{76}$$

The dimensions of the submatrices of M_1 are given by $M_1^{b,1}, M_1^{b,2} \in \mathbb{R}^{N_p\times 3N_p}$ and $M_1^d \in \mathbb{R}^{N_p\times 9N_p}$. The submatrices of M_1 can be found somewhat further in the report. For M_2, the submatrices all have same size $M_2^{b,1}, M_2^{b,2}, M_2^d \in \mathbb{R}^{N_p\times 1}$. Finally, for M_3, the same holds, thus $M_3^{b,1}, M_3^{b,2}, M_3^d \in \mathbb{R}^{N_p\times 1}$. The submatrices are defined as:

$$M_2^{b,1} = \begin{bmatrix} (A^{b,1}) \\ (A^{b,1})^2 \\ (A^{b,1})^3 \\ \vdots \\ (A^{b,1})^{(l+1)} \\ \vdots \\ (A^{b,1})^{N_p} \end{bmatrix} \qquad M_2^{b,2} = \begin{bmatrix} (A^{b,2}) \\ (A^{b,2})^2 \\ (A^{b,2})^3 \\ \vdots \\ (A^{b,2})^{(l+1)} \\ \vdots \\ (A^{b,2})^{N_p} \end{bmatrix} \qquad M_2^d = \begin{bmatrix} (A^d) \\ (A^d)^2 \\ (A^d)^3 \\ \vdots \\ (A^d)^{(l+1)} \\ \vdots \\ (A^d)^{N_p} \end{bmatrix} \tag{77}$$

$$M_1^d(\hat{\delta}_d) = \begin{bmatrix} B_2^d & 0 & 0 & \dots & 0 \\ (A^d)B_2^d & B_2^d & 0 & \dots & 0 \\ (A^d)^2B_2^d & (A^d)B_2^d & B_2^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (A^d)^lB_2^d & (A^d)^{(l-1)}B_2^d & (A^d)^{(l-2)}B_2^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ (A^d)^{(N_p-1)}B_2^d & (A^d)^{(N_p-2)}B_2^d & (A^d)^{(N_p-3)}B_2^d & \dots & B_2^d \end{bmatrix} \tag{78}$$

$$M_1^d(u_d) = \begin{bmatrix} B_1^d & 0 & 0 & \dots & 0 \\ (A^d)B_1^d & B_1^d & 0 & \dots & 0 \\ (A^d)^2B_1^d & (A^d)B_1^d & B_1^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (A^d)^lB_1^d & (A^d)^{(l-1)}B_1^d & (A^d)^{(l-2)}B_1^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ (A^d)^{(N_p-1)}B_1^d & (A^d)^{(N_p-2)}B_1^d & (A^d)^{(N_p-3)}B_1^d & \dots & B_1^d \end{bmatrix} \tag{79}$$

$$M_1^d(\hat{z}_d) = \begin{bmatrix} B_3^d & 0 & 0 & \dots & 0 \\ (A^d)B_3^d & B_3^d & 0 & \dots & 0 \\ (A^d)^2B_3^d & (A^d)B_3^d & B_3^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (A^d)^lB_3^d & (A^d)^{(l-1)}B_3^d & (A^d)^{(l-2)}B_3^d & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ (A^d)^{(N_p-1)}B_3^d & (A^d)^{(N_p-2)}B_3^d & (A^d)^{(N_p-3)}B_3^d & \dots & B_3^d \end{bmatrix} \tag{80}$$

$$
M_1^{b,1} = \left[\begin{array}{ccc|cccccc|cccccc}
0 & \dots & 0 & B_1^{b,1} & 0 & \dots & & & 0 & B_3^{b,1} & 0 & \dots & & & 0 \\
0 & \dots & 0 & (A^{b,1})B_1^{b,1} & B_1^{b,1} & 0 & \dots & & 0 & (A^{b,1})B_3^{b,1} & B_3^{b,1} & 0 & \dots & & 0 \\
\vdots & \ddots & \vdots & (A^{b,1})^2B_1^{b,1} & (A^{b,1})B_1^{b,1} & B_1^{b,1} & \ddots & & 0 & (A^{b,1})^2B_3^{b,1} & (A^{b,1})B_3^{b,1} & B_3^{b,1} & 0 & \dots & 0 \\
\vdots & \ddots & & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\
0 & \dots & 0 & (A^{b,1})^lB_1^{b,1} & (A^{b,1})^{(l-1)}B_1^{b,1} & (A^{b,1})^{(l-2)}B_1^{b,1} & \dots & & 0 & (A_1^{b,1})^lB_3b,1 & (A_1^{b,1})^{(l-1)}B_3b,1 & (A_1^{b,1})^{(l-2)}B_3b,1 & \dots & & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \dots & 0 \\
0 & \dots & 0 & (A^{b,1})^{(N_p-1)}B_1^{b,1} & (A^{b,1})^{(N_p-2)}B_1^{b,1} & (A^{b,1})^{(N_p-3)}B_1^{b,1} & \dots & (A^{b,1})B_1^{b,1} & B_1^{b,1} & (A^{b,1})^{(N_p-1)}B_3^{b,1} & (A^{b,1})^{(N_p-2)}B_3^{b,1} & (A^{b,1})^{(N_p-3)}B_3^{b,1} & \dots & & B_3^{b,1}
\end{array}\right] \tag{81}
$$

$$
M_1^{b,2} = \left[\begin{array}{ccc|cccccc|cccccc}
0 & \dots & 0 & B_1^{b,2} & 0 & \dots & & & 0 & B_3^{b,2} & 0 & \dots & & & 0 \\
0 & \dots & 0 & (A^{b,2})B_1^{b,2} & B_1^{b,2} & 0 & \dots & & 0 & (A^{b,2})B_3^{b,1} & B_3^{b,2} & 0 & \dots & & 0 \\
\vdots & \ddots & \vdots & (A^{b,2})^2B_1^{b,2} & (A^{b,2})B_1^{b,2} & B_1^{b,2} & \ddots & & 0 & (A^{b,2})^2B_3^{b,2} & (A^{b,2})B_3^{b,2} & B_3^{b,2} & 0 & \dots & 0 \\
\vdots & \ddots & & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\
0 & \dots & 0 & (A^{b,2})^lB_1^{b,2} & (A^{b,2})^{(l-1)}B_1^{b,2} & (A^{b,2})^{(l-2)}B_1^{b,2} & \dots & & 0 & (A_1^{b,2})^lB_3b,2 & (A_1^{b,2})^{(l-1)}B_3b,2 & (A_1^{b,2})^{(l-2)}B_3b,2 & \dots & & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \dots & 0 \\
0 & \dots & 0 & (A^{b,2})^{(N_p-1)}B_1^{b,2} & (A^{b,2})^{(N_p-2)}B_1^{b,2} & (A^{b,2})^{(N_p-3)}B_1^{b,2} & \dots & (A^{b,2})B_1^{b,2} & B_1^{b,2} & (A^{b,2})^{(N_p-1)}B_3^{b,2} & (A^{b,2})^{(N_p-2)}B_3^{b,2} & (A^{b,2})^{(N_p-3)}B_3^{b,2} & \dots & & B_3^{b,2}
\end{array}\right] \tag{82}
$$

$$M_3^{b,1} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \qquad M_3^{b,2} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \qquad M_3^{d} = \begin{bmatrix} B_4^d \\ B_4^d(1 + A^d) \\ B_4^d(1 + A^d + (A^d)^2) \\ \vdots \\ B_4^d(1 + A^d + (A^d)^2 + ... + (A^d)^{N_p-1}) \end{bmatrix} \tag{83}$$

Constraint equations MILP

The constraint equations, given by 68 and 70 for the batteries and diesel generator respectively, can be summarized as follows:

$$F_1 \tilde{V}(k) \leq F_2 + F_3 x(k) \tag{84}$$

Here, the following holds for the matrices:

$$F_1 = \begin{bmatrix} F_1^{b,1} & 0 & 0 \\ 0 & F_1^{b,2} & 0 \\ 0 & 0 & F_1^d \end{bmatrix} \qquad F_2 = \begin{bmatrix} F_2^{b,1} \\ F_2^{b,2} \\ F_2^d \end{bmatrix} \qquad F_3 = \begin{bmatrix} F_3^{b,1} & 0 & 0 \\ 0 & F_3^{b,2} & 0 \\ 0 & 0 & F_3^d \end{bmatrix}$$

The dimensions of the submatrices of F_1 are given by $F_1^{b,1}, F_1^{b,2} \in \mathbb{R}^{8N_p \times 3N_p}$ and $F_1^d \in \mathbb{R}^{27N_p \times 9N_p}$. For F_2, the submatrices are of size $F_2^{b,1}, F_2^{b,2} \in \mathbb{R}^{8N_p \times 1}$ and $F_2^d \in \mathbb{R}^{27N_p \times 1}$. Finally, for F_3: $F_3^{b,1}, F_3^{b,2} \in \mathbb{R}^{8N_p \times 1}$ and $F_3^d \in \mathbb{R}^{27N_p \times 1}$. The submatrices are defined as:

$$F_2^{b,1} = \begin{bmatrix} g_5^{b,1} \\ \vdots \\ g_5^{b,1} \end{bmatrix} \qquad F_2^{b,2} = \begin{bmatrix} g_5^{b,2} \\ \vdots \\ g_5^{b,2} \end{bmatrix} \qquad F_2^d = \begin{bmatrix} g_5^d \\ \vdots \\ g_5^d \end{bmatrix} - \begin{bmatrix} 0 \\ E_1^d B_4^d \\ E_1^d B_4^d(1 + A^d) \\ E_1^d B_4^d(1 + A^d + (A^d)^2) \\ \vdots \\ E_1^d B_4^d(1 + A^d + (A^d)^2 + ... + (A^d)^{N_p-2}) \end{bmatrix} \tag{85}$$

$$F_3^{b,1} = \begin{bmatrix} -E_1^{b,1} \\ -E_1^{b,1} A^{b,1} \\ -E_1^{b,1} (A^{b,1})^2 \\ \vdots \\ -E_1^{b,1} (A^{b,1})^{N_p-1} \end{bmatrix} \qquad F_3^{b,2} = \begin{bmatrix} -E_1^{b,2} \\ -E_1^{b,2} A^{b,2} \\ -E_1^{b,2} (A^{b,2})^2 \\ \vdots \\ -E_1^{b,2} (A^{b,2})^{N_p-1} \end{bmatrix} \qquad F_3^d = \begin{bmatrix} -E_1^d \\ -E_1^d A^d \\ -E_1^d (A^d)^2 \\ \vdots \\ -E_1^d (A^d)^{N_p-1} \end{bmatrix} \tag{86}$$

$$F_1^{b,1} = \left[\begin{array}{cccccc|cccc|cccc}
E_3^{b,1} & 0 & 0 & 0 & \dots & 0 & E_2^{b,1} & 0 & 0 & \dots & 0 & E_4^{b,1} & 0 & 0 & \dots & 0 \\
0 & E_3^{b,1} & 0 & 0 & \dots & 0 & E_1^{b,1}B_1^{b,1} & E_2^{b,1} & 0 & \dots & 0 & E_1^{b,1}B_3^{b,1} & E_4^{b,1} & 0 & \dots & 0 \\
0 & 0 & E_3^{b,1} & 0 & \dots & 0 & E_1^{b,1}A^{b,1}B_1^{b,1} & E_1^{b,1}B_1^{b,1} & E_2^{b,1} & \dots & 0 & E_1^{b,1}A^{b,1}B_3^{b,1} & E_1^{b,1}B_3^{b,1} & E_4^{b,1} & \dots & 0 \\
\vdots & \vdots & & \ddots & & & \vdots & \vdots & & \ddots & \vdots & \vdots & & & \ddots & \\
0 & 0 & \dots & & & E_3^{b,1} & E_1^{b,1}(A^{b,1})^{N_p-2}B_1^{b,1} & E_1^{b,1}(A^{b,1})^{N_p-3}B_1^{b,1} & & & E_2^{b,1} & E_1^{b,1}(A^{b,1})^{N_p-2}B_3^{b,1} & E_1^{b,1}(A^{b,1})^{N_p-3}B_3^{b,1} & & & E_4^{b,1}
\end{array}\right] \tag{87}$$

$$F_1^{b,2} = \left[\begin{array}{cccccc|cccc|cccc}
E_3^{b,2} & 0 & 0 & 0 & \dots & 0 & E_2^{b,2} & 0 & 0 & \dots & 0 & E_4^{b,2} & 0 & 0 & \dots & 0 \\
0 & E_3^{b,2} & 0 & 0 & \dots & 0 & E_1^{b,2}B_1^{b,2} & E_2^{b,2} & 0 & \dots & 0 & E_1^{b,2}B_3^{b,2} & E_4^{b,2} & 0 & \dots & 0 \\
0 & 0 & E_3^{b,2} & 0 & \dots & 0 & E_1^{b,2}A^{b,2}B_1^{b,2} & E_1^{b,2}B_1^{b,2} & E_2^{b,2} & \dots & 0 & E_1^{b,2}A^{b,2}B_3^{b,2} & E_1^{b,2}B_3^{b,2} & E_4^{b,2} & \dots & 0 \\
\vdots & \vdots & & \ddots & & & \vdots & \vdots & & \ddots & \vdots & \vdots & & & \ddots & \\
0 & 0 & \dots & & & E_3^{b,2} & E_1^{b,2}(A^{b,2})^{N_p-2}B_1^{b,2} & E_1^{b,2}(A^{b,2})^{N_p-3}B_1^{b,2} & & & E_2^{b,2} & E_1^{b,2}(A^{b,2})^{N_p-2}B_3^{b,2} & E_1^{b,2}(A^{b,2})^{N_p-3}B_3^{b,2} & & & E_4^{b,2}
\end{array}\right] \tag{88}$$

$$F_1^{d} = \left[\begin{array}{cccccc|ccccc|ccccc}
E_3^{d} & 0 & 0 & 0 & \dots & 0 & E_2^{d} & 0 & 0 & \dots & 0 & E_4^{d} & 0 & 0 & \dots & 0 \\
E_1^{d}B_2^{d} & E_3^{d} & 0 & 0 & \dots & 0 & 0 & E_2^{d} & 0 & \dots & 0 & E_1^{d}B_3^{d} & E_4^{d} & 0 & \dots & 0 \\
E_1^{d}A^{d}B_2^{d} & E_1^{d}B_2^{d} & E_3^{d} & 0 & \dots & 0 & 0 & 0 & E_2^{d} & \dots & 0 & E_1^{d}A^{d}B_3^{d} & E_1^{d}B_3^{d} & E_4^{d} & \dots & 0 \\
\vdots & \vdots & & \ddots & & & \vdots & \vdots & & \ddots & \vdots & \vdots & & & \ddots & \\
E_1^{d}(A^{d})^{N_p-2}B_2^{d} & E_1^{d}(A^{d})^{N_p-3}B_2^{d} & & & & E_3^{d} & 0 & 0 & & \dots & E_2^{d} & E_1^{d}(A^{d})^{N_p-2}B_3^{d} & E_1^{d}(A^{d})^{N_p-3}B_3^{d} & & & E_4^{d}
\end{array}\right] \tag{89}$$

Cost function

The cost function of the microgrid at time step k is given by:

$$J(k) = \sum_{j=0}^{N_\mathrm{p}-1} \left(\sum_{i=1}^{N_\mathrm{b}} W_{\mathrm{b},i} \left| \Delta s_{\mathrm{b},i}(k+j) \right| + W_\mathrm{d} \left| \Delta s_\mathrm{d}(k+j) \right| \right) - W_{\text{ fuel}} \Big(x_\mathrm{d} \left(k + N_\mathrm{p}\right) - x_\mathrm{d}(k) \Big) \tag{90}$$

$$- W_\mathrm{e} \sum_{i=1}^{N_\mathrm{b}} \Big(x_{\mathrm{b},i} \left(k + N_\mathrm{p}\right) - x_{\mathrm{b},i}(k) \Big) + \sum_{j=0}^{N_\mathrm{p}-1} P_\mathrm{imp}(k+j) C_\mathrm{e}(k+j) \tag{91}$$

Here, $P_{imp}(k)$ is defined by:

$$P_\mathrm{imp}(k+j) = P_\mathrm{load}(k+j) - u_\mathrm{d}(k+j) - \sum_{i=1}^{N_\mathrm{b}} u_{\mathrm{b},i}(k+j), \quad \forall j \tag{92}$$

The values of $\tilde{C}_e, \tilde{P}_{\text{ load }}$ are known and given by the vectors:

$$\tilde{C}_\mathrm{e}(k) = \left[\begin{array}{lll} C_\mathrm{e}(k) & \ldots & C_\mathrm{e}\left(k + N_\mathrm{p} - 1\right) \end{array} \right]^T$$

$$\tilde{P}_{\text{ load }}(k) = \left[\begin{array}{lll} P_{\text{ load }}(k) & \ldots & P_{\text{ load }}\left(k + N_\mathrm{p} - 1\right) \end{array} \right]^T$$

The cost function can be rewritten in terms of the estimate vector $\tilde{V}(k)$ (given by equations 72,73), $\tilde{x}(k)$, $x(k)$ (given by equations 74) and vectors $\tilde{C}_e, \tilde{P}_{\text{ imp }}$:

$$J(k) = W_1 | W_2 \tilde{V}(k) | + W_3 \tilde{x}(k) + W_4 x(k) + \sum_{j=0}^{N_\mathrm{p}-1} P_\mathrm{imp}(k+j) C_\mathrm{e}(k+j) \tag{93}$$

Here, the weights W_1 to W_4, with the exception of W_2, are defined as follows:

$$W_1 = \left[\begin{array}{cccc|cccc|cccc} W_{b,1} & \ldots & W_{b,1} & 0 & W_{b,2} & \ldots & W_{b,2} & 0 & W_d & \ldots & W_d & 0 \end{array} \right] \tag{94}$$

$$W_3 = \left[\begin{array}{cccc|cccc|cccc} 0 & \ldots & 0 & -W_e & 0 & \ldots & 0 & -W_e & 0 & \ldots & 0 & -W_\mathrm{fuel} \end{array} \right] \tag{95}$$

$$W_4 = \left[\begin{array}{ccc} W_e & W_e & W_\mathrm{fuel} \end{array} \right] \tag{96}$$

For $W_1 \in \mathbb{R}^{1 \times 3N_p}$, $W_3 \in \mathbb{R}^{1 \times 3N_p}$ and $W_4 \in \mathbb{R}^{1 \times 3}$.

For W_2, the following holds:

$$W_2 = \begin{bmatrix} W_2^{b,1} & 0 & 0 \\ 0 & W_2^{b,2} & 0 \\ 0 & 0 & W_2^d \end{bmatrix} \tag{97}$$

$$W_2^{b,1} = \left[\begin{array}{cccc|ccc|ccc} 1 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ -1 & 1 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ & \ddots & \ddots & & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \ldots & -1 & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{array} \right] \quad W_2^{b,2} = \left[\begin{array}{cccc|ccc|ccc} 1 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ -1 & 1 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ & \ddots & \ddots & & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \ldots & -1 & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{array} \right] \tag{98}$$

$$W_2^d = \left[\begin{array}{ccccccccccccc|ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & \ldots & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & & \ldots & & -1 & -1 & -1 & -1 & & 1 & 1 & 1 & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{array} \right] \tag{99}$$

Here, $W_2 \in \mathbb{R}^{3N_p \times 15N_p}$ and the submatrices are $W_2^{b,1}, W_2^{b,2} \in \mathbb{R}^{N_p \times 3N_p}$ and $W_2^d \in \mathbb{R}^{N_p \times 9N_p}$.

Next, the estimate vector $\tilde{V}(k)$ is expanded with term P_{imp} as follows:

$$\tilde{V}_{\text{new}}(k) = \begin{bmatrix} \tilde{V}_{b,1}(k) \\ \tilde{V}_{b,2}(k) \\ \tilde{V}_d(k) \\ P_{\text{imp}}(k) \\ \vdots \\ P_{\text{imp}}(k+N_p) \end{bmatrix} \tag{100}$$

The submatrices $\tilde{V}_{b,1}(k), \tilde{V}_{b,2}(k), \tilde{V}_d(k)$ are still given by equation 73.

Substituting the update equation found for $\tilde{x}(k)$ and using the new defined vector $\tilde{V}_{\text{new}}(k)$, the cost function expression 93 can be simplified to:

$$\begin{aligned} J(k) &= W_1|W_2\tilde{V}(k)| + W_3(M_1\tilde{V}(k) + M_2x(k) + M_3) + W_4x(k) + \sum_{j=0}^{N_\text{p}-1} P_{\text{imp}}(k+j)C_{\text{e}}(k+j) \\ &= W_1|W_2\tilde{V}(k)| + W_3M_1\tilde{V}(k) + (W_3M_2 + W_4)x(k) + W_3M_3 + \sum_{j=0}^{N_\text{p}-1} P_{\text{imp}}(k+j)C_{\text{e}}(k+j) \\ &= W_1|W_{2,\text{new}}\tilde{V}_{\text{new}}(k)| + S_1\tilde{V}_{\text{new}}(k) + S_2x(k) + W_3M_3 \end{aligned}$$

The new matrices are defined as:

$$W_{2,\text{new}} = \left[\begin{array}{ccc|ccc} W_2^{b,1} & 0 & 0 & 0 & \dots & 0 \\ 0 & W_2^{b,2} & 0 & 0 & \dots & 0 \\ 0 & 0 & W_2^d & 0 & \dots & 0 \end{array}\right] \tag{101}$$

The zero-matrix added is of size $3N_p \times N_p$ and is multiplied with the P_{imp} terms in the $\tilde{V}_{\text{new}}(k)$ vector. Thus, $W_2 \in \mathbb{R}^{3N_p \times 16N_p}$. The two S matrices are given by:

$$S_1 = \begin{bmatrix} W_3M_1 & \tilde{C}_e(k) & 0 \end{bmatrix} \quad S_2 = W_3M_2 + W_4 \tag{102}$$

Here, $S_1 \in \mathbb{R}^{1\times 16N_p}$ and $S_2 \in \mathbb{R}^{1\times 3}$.

As a final step, the definition of P_{imp} given by equation 92 should be added to the constraints as follows:

$$\begin{aligned} P_{\text{imp}}(k+j) &\leq P_{\text{load}}(k+j) - u_{\text{d}}(k+j) - \sum_{i=1}^{N_\text{b}} u_{\text{b},i}(k+j), \quad \forall j \\ P_{\text{imp}}(k+j) &\geq P_{\text{load}}(k+j) - u_{\text{d}}(k+j) - \sum_{i=1}^{N_\text{b}} u_{\text{b},i}(k+j), \quad \forall j \end{aligned}$$

Rewriting these constraints:

$$u_{\text{d}}(k+j) + \sum_{i=1}^{N_\text{b}} u_{\text{b},i}(k+j) + P_{\text{imp}}(k+j) \leq P_{\text{load}}(k+j), \quad \forall j \tag{103}$$

$$-u_{\text{d}}(k+j) - \sum_{i=1}^{N_\text{b}} u_{\text{b},i}(k+j) - P_{\text{imp}}(k+j) \leq -P_{\text{load}}(k+j), \quad \forall j \tag{104}$$

These two constraints are added to the general inequality expression to obtain a new constraint equation:

$$F_{1,\text{new}}\tilde{V}_{\text{new}}(k) \leq F_{2,\text{new}} + F_{3,\text{new}}x(k) \tag{105}$$

$$F_{1,new} = \left[\begin{array}{ccc|c} F_1^{b,1} & 0 & 0 & 0 \ \dots \ 0 \\ 0 & F_1^{b,2} & 0 & 0 \ \dots \ 0 \\ 0 & 0 & F_1^d & 0 \ \dots \ 0 \\ \hline F_{11}^{b,1} & F_{11}^{b,2} & F_{11}^d & I \\ -F_{11}^{b,1} & -F_{11}^{b,2} & -F_{11}^d & -I \end{array}\right] \quad F_{2,new} = \left[\begin{array}{c} F_2^{b,1} \\ F_2^{b,2} \\ F_2^d \\ \hline P_{\text{load}}(k) \\ \vdots \\ P_{\text{load}}(k+N_p) \\ -P_{\text{load}}(k) \\ \vdots \\ -P_{\text{load}}(k+N_p) \end{array}\right] \quad F_{3,new} = \left[\begin{array}{ccc} F_3^{b,1} & 0 & 0 \\ 0 & F_3^{b,2} & 0 \\ 0 & 0 & F_3^d \\ \hline 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{array}\right] \tag{106}$$

The dimensions of the submatrices in the left upper corner of $F_{1,new}$ are given by $43N_p \times 15N_p$, the zeros in the right upper corner by $43N_p \times N_p$, the matrix in the corner left down by $2N_p \times 15N_p$ and finally, the identity matrices are both $N_p \times N_p$.
For $F_{2,new}$, the upper half is of size $43N_p \times 1$. The $+P_{\text{load}}$ and $-P_{\text{load}}$ vectors are both N_p long.
Finally, for $F_{3,new}$, the upper half is of size $43N_p \times 3$ and the zero matrix beneath it has dimension $2N_p \times 3$.
The submatrices F_{11} located in the $F_{1,\text{new}}$ matrix, for $F_{11}^{b,1} = F_{11}^{b,2} \in \mathbb{R}^{N_p \times 9N_p}$ and $F_{11}^d \in \mathbb{R}^{N_p \times 15N_p}$, are given by:

$$F_{11}^{b,1} = F_{11}^{b,2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ & & & & & & & \ddots & \ddots & \ddots & & \\ 0 & & & & \dots & & & & 0 & 0 & 1 & 0 \end{bmatrix} \tag{107}$$

$$F_{11}^d = \begin{bmatrix} 0&0&0&0&1&0&0&0&0 & 0&0&0&0&0&0&0&0&0 & & & & & & & & & 0 \ \dots \ 0 \\ 0&0&0&0&0&0&0&0&0 & 0&0&0&0&1&0&0&0&0 & & & & & & & & & 0 \ \dots \ 0 \\ & & & & & & & & & \ddots&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots & & & & & & & & & \\ 0 & & & & & & & & \dots & & & & & & & & & 0 & 0&0&0&0&1&0&0&0&0 \end{bmatrix} \tag{108}$$

When minimizing this cost function, subjected to the constraints as defined above, the absolute value lines should be removed. This is done by making use of the following proposition from the course *Optimization in Systems and Control*:

> Proposition 8.2 Consider $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$. If $w_i > 0$ for all i, then the following problems are equivalent:
>
> $$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{n} w_i \left| x_i \right| \text{ subject to } Ax \leqslant b$$
>
> $$\min_{x, \alpha \in \mathbb{R}^n} \sum_{i=1}^{n} w_i \alpha_i \text{ subject to } Ax \leqslant b, \alpha \geqslant x \text{ and } \alpha \geqslant -x$$

With this proposition, the first part of the cost function can be rewritten. Note that the sizes of the vectors are $\tilde{V}_{\text{new}}(k) \in \mathbb{R}^{16N_p \times 1}$, $\tilde{H}(k) \in \mathbb{R}^{3N_p \times 1}$, $x(k) \in \mathbb{R}^{3 \times 1}$.

$$\begin{aligned} &\min_{\tilde{V}_{\text{new}}(k)} \ W_1 | W_{2,\text{new}} \tilde{V}_{\text{new}}(k) | \\ &\text{subject to} \\ &F_{1,\text{new}} \tilde{V}_{\text{new}}(k) \leq F_{2,\text{new}} + F_{3,\text{new}} x(k) \end{aligned}$$

Equals:

$$\begin{aligned} &\min_{\tilde{V}_{\text{new}}(k), \tilde{H}(k)} \ W_1 \tilde{H}(k) \\ &\text{subject to} \\ &F_{1,\text{new}} \tilde{V}_{\text{new}}(k) \leq F_{2,\text{new}} + F_{3,\text{new}} x(k) \\ &-\tilde{H}(k) \leq W_{2,\text{new}} \tilde{V}_{\text{new}}(k) \leq \tilde{H}(k) \end{aligned}$$

For the final minimization problem, the constant term $W_3 M_3$ is omitted, as well as $S_2 x(k)$. The current states $x(k)$ do not appear in the cost function since it is a known value at time step k. Therefore, it is not an optimization variable. $x(k)$ does appear in the inequality constraints, since the constraints on the optimization variables depend on the current state. With the extra constraints added and the constant terms omitted, the final minimization problem becomes:

$$\begin{aligned}
&\min_{\tilde{V}_{\text{new}}(k), \tilde{H}(k)} \quad W_1 \tilde{H}(k) + S_1 \tilde{V}_{\text{new}}(k) \\
&\text{subject to} \\
&F_{1,\text{new}} \tilde{V}_{\text{new}}(k) \leq F_{2,\text{new}} + F_{3,\text{new}} x(k) \\
&-\tilde{H}(k) \leq W_{2,\text{new}} \tilde{V}_{\text{new}}(k) \leq \tilde{H}(k)
\end{aligned} \tag{109}$$

Result

For both of the batteries and the diesel generator, the following result is found for the optimal states:

	Battery 1	**Battery 2**
$\hat{\delta}_b$ **[-]**	0	0
u_b **[kW]**	0	0
$\hat{z}_b$ **[kW]**	1	0

	Diesel generator
$\hat{\delta}_d$ **[-]**	$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$
u_d **[kW]**	0
$\hat{z}_d$ **[kW]**	$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$

All $\hat{\delta}$ are equal to zero, which means that both of the batteries are on operational mode discharging, and the diesel generator is on operational mode off. As a consequence, the exchanged power of the batteries is 0 kW and the generated power by the diesel generator is 0 kW as well. $\hat{z}$ gives an odd result for battery 1 and for the diesel generator: both values contain a 1. Since z is defined as $u\delta$ and δ is 0 for both batteries and the diesel generator, z can never become 1. Therefore, it can be concluded that the optimization algorithm with the constraints and cost function as defined before, does not perform as it should.

Step 2.9

In this section, the closed-loop behaviour of the system is simulated using the receding horizon approach. This means that at each time step k the optimal MPC control input will be computed following the steps of section 2.8. The result of the MPC optimalization procedure at time step k is the optimal vector as given by 72.

At the time instant k, the $\delta_{b,1}(k)$, $\delta_{b,2}(k)$, $\delta_d(k)$, $u_{b,1}(k)$, $u_{b,2}(k)$, $u_d(k)$ and $z_{b,1}(k)$, $z_{b,2}(k)$ and $z_d(k)$ from 72 will be implemented in the update equations for the batteries and diesel generator 53, 54 and 55. Hereafter, the optimal MPC control input is recalculated for the next time-step $k+1$ with the updated states $x_{b,1}$, $x_{b,2}$ and x_d in the optimization problem. This procedure can be repeated up until the desired time-step.

The results, plotted over 36 hours, are shown below in figures 6 and 7. States $x_{b,1}$ and $x_{b,2}$ only decrease with a small amount. State x_d decreases at first, and then switches between two values close to each other. This is not an optimal result. Furthermore, figure 7 shows no control input for all time steps. Thus, it can be concluded no optimal solution is found and that there is an issue with the code.

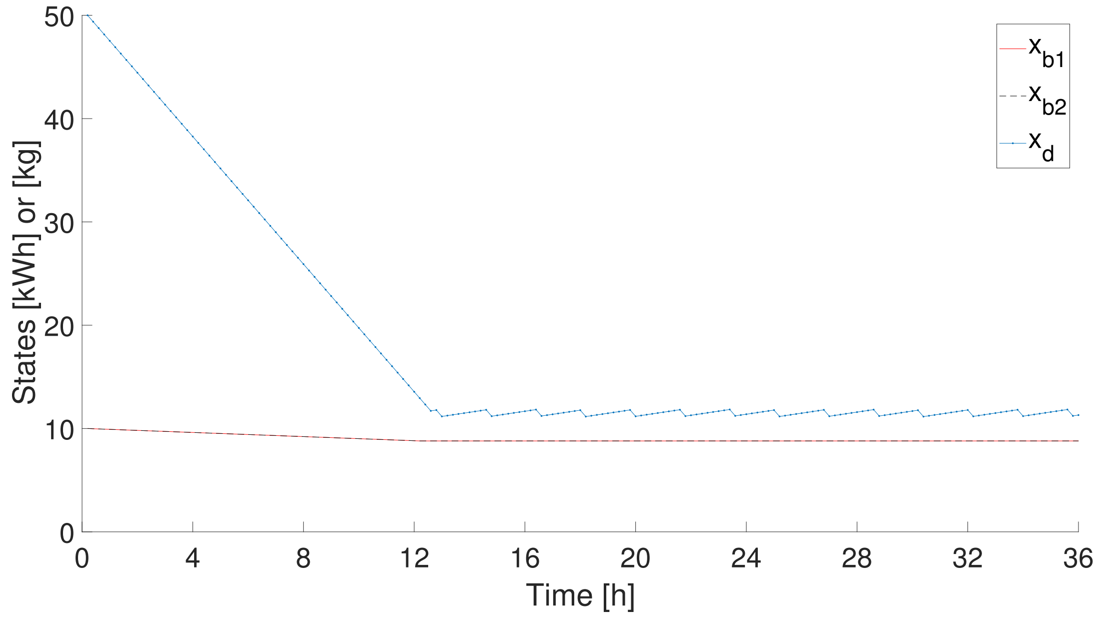

Figure 6: States $x_{b,1}$ [kWh] (red line) $x_{b,2}$ [kWh] (black striped line) and x_d [kg] (blue dotted line), plotted against time [h] for 36 hours

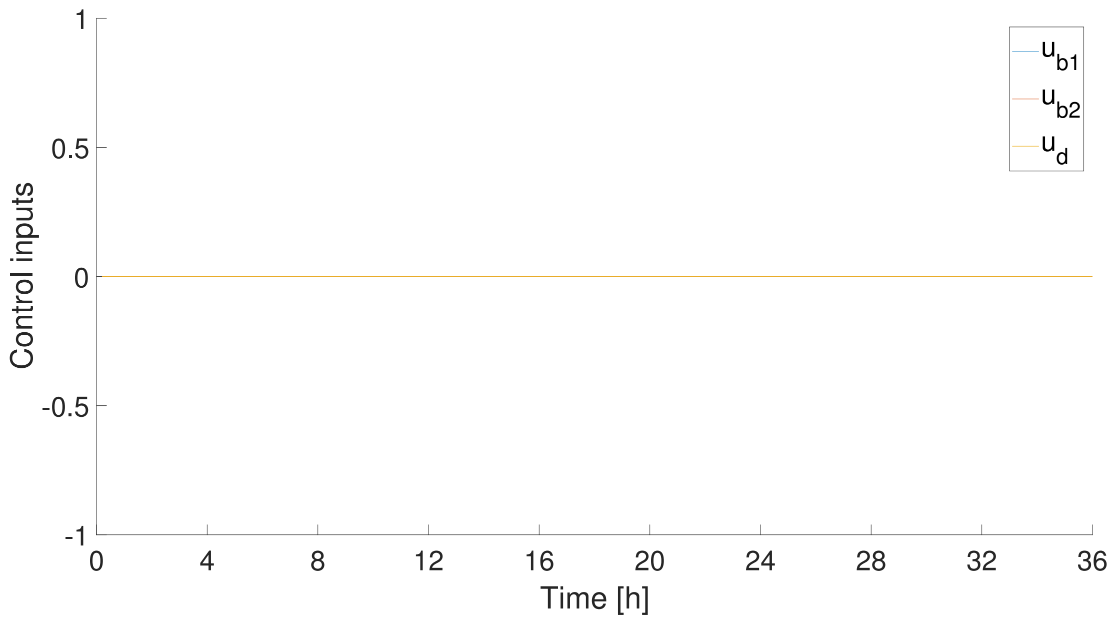

Figure 7: Control inputs $u_{b,1}, u_{b,2}, u_d$ [kW], plotted against time [h] for 36 hours

Step 2.10

The time step T_s is assumed to be 0.20 [h], as defined in step 2.7. Furthermore, it is assumed that at time step $x(1)$ the time is 0.20 hours after 12AM, and thus initial value x_0 starts at 12AM. There is a constraint added on the batteries every time it is either 12AM or 12PM. Therefore, for every $k = 60, 120, 180, ...$, the constraint on $x_{b,1}$ and $x_{b,2}$ becomes as follows:

$$x_{b,i} \geq 0.2\overline{x}_{b,i} \quad \longrightarrow \quad -x_{b,i} \leq -0.2\overline{x}_{b,i} \tag{110}$$

Which equals the following additions to the constraint matrices:

$$E_1^{b,i,\text{addition}} = -1 \quad E_2^{b,i,\text{addition}} = 0 \quad E_3^{b,i,\text{addition}} = 0 \quad E_4^{b,i,\text{addition}} = 0 \quad g_5^{b,i,\text{addition}} = -0.2\overline{x}_d \tag{111}$$

The algorithm in MATLAB is the same as in 2.7, however, it is expanded with an if-loop such that whenever $k = 60, 120, 180, ...$, the extra constraints are added.

MATLAB gives an infeasible result. Constraint 110 can never be met: $\overline{x}_{b,1} = 48$ and $\overline{x}_{b,2} = 64$, thus $0.2\overline{x}_{b,1} = 9.6$ and $0.2\overline{x}_{b,2} = 12.8$. Before time step $k = 60$ is reached, the states are already below these lower bounds, see figure 6. Therefore, MATLAB can not give a feasible result. For this reason, in Appendix **??**, the piece of code that should be implemented for this exercise is commented out.

Conclusion

While working on this assignment, several skills were learned and insights obtained. Also, several aspects of the implementation could be done differently if this assignment would be done again. This will be briefly outlined below.

Insights

The first main insight is the way to use an optimizer for Mixed Integer Linear Programming (MILP) problems. For Systems & Control, optimization and minimization algorithms have been worked with before, in the course *Optimization in Systems and Control* for example. However, the optimizer *Gurobi* was never used, which works with MILP problems. Rewriting a PWA into a MILP problem required a new approach to setting up constraints and with this new approach, knowledge was gained. The fact that there were many constraints, made it an insightful job to set them up, check them, and check if there might be two constraints that are equal, such that only one of them is needed. Furthermore, the size of the matrices in the update equations, inequality constraints and weights in the cost function was extremely large due to the many constraints and the prediction horizon. Therefore, stacking of submatrices was needed. This required very precise definition of all matrices, both on paper as well as in MATLAB. A lot of consistency was needed when checking the values and variables in the matrices - finding a smart way to check the matrices on paper with the MATLAB matrices and checking the matrices on errors, was also informative.

Another main insight that was obtained during this assignment, is how to use binary variables in describing system dynamics. The description of system dynamics in the piecewise affine (PWA) form is quite straightforward, however, to rewrite the equations to the mixed logical dynamic (MLD) form, binary variables arise to be able to describe the switching between the system modes, where each mode exhibits different dynamical behaviour. Along with the binary variables, many constraints arise, to assign when the binary variables should be turned on or off. The rewriting of these constraints so that they can be used in a linear programming algorithm proved to be a fundamental point of the exercise.

The last main insight that will be highlighted is the size of the system description that was obtained during the assignment. The microgrid consists of only 3 subsystems; the diesel generator and the two batteries. In the full system description of the microgrid, as stated above, many large matrices were obtained. 11 matrices arise in the MLD update equations for the subsystems and 15 matrices arise in the MLD constraint equations. Furthermore, for the implementation int he optimization algorithm another 7 matrices emerged, each of which consisted of multiple matrices, of which the submatrices again are made up of the multiple MLD matrices. When the systems and matrices are described and noted clearly, everything stays comprehensible. However, many different hybrid system model types exist and it could be interesting to consider other hybrid system descriptions whenever the system includes more subsystems or the subsystems behave according to more complex switching patterns.

Discussion

The first time the MATLAB code ran for step 2.8, it gave infinite bounds and thus an infeasible solution. For a few days, multiple methods were tried and an attempt was made to find mistakes in either the code or in the stacking of the matrices. Now, the code does run, but the solution it gives is not a logical one. It is not clear what the issue is exactly. Although the attempts to solve the problem were very insightful, because mistakes were found, for a next time, it would be recommended to rewrite the code from the start instead of looking for mistakes. Also, the model formulation could be adjusted, such as defining vectors differently or adding extra constraints, such that the system might work.

Appendix

Exercise 2.3

```
%% SC4160 MODELLING AND CONTROL OF HYBRID SYSTEMS
% Step 2.3
% Jessie van Dam (4395832) and Miranda van Duijn (4355776)
clear all; close all; clc;

% Region 1
syms a1 b1 ud1
fun11 = (ud1^2 + 4 - a1 - b1*ud1).^2;
fun12 = (4*ud1 - a1 - b1*ud1).^2;
int11 = int(fun11,ud1,0,2);
int12 = int(fun12,ud1,2,5);
g1 = matlabFunction(int11+int12);

pdiff1a = diff(g1,a1);
pdiff1b = diff(g1,b1);

sol1 = solve(pdiff1a,pdiff1b);

parD.a1 = double(sol1.a1);
parD.b1 = double(sol1.b1);

% Region 2
syms a2 b2 ud2
fun2 = (-9.44*ud2^3 + 166.06*ud2^2 -948.22*ud2 + 1790.28 - a2 - b2*ud2).^2;
int2 = int(fun2,ud2,5,6.5);
g2 = matlabFunction(int2);

pdiff2a = diff(g2,a2);
pdiff2b = diff(g2,b2);

sol2 = solve(pdiff2a,pdiff2b);

parD.a2 = double(sol2.a2);
parD.b2 = double(sol2.b2);

% Region 3
syms a3 b3 ud3
fun31 = (-9.44*ud3^3 + 166.06*ud3^2 -948.22*ud3 + 1790.28 - a3 - b3*ud3).^2;
fun32 = (-11.78*ud3 + 132.44 - a3 - b3*ud3).^2;
fun33 = (4.01*(ud3-10.47).^2 + 17.79 - a3 - b3*ud3).^2;
int31 = int(fun31,ud3,6.5,7);
int32 = int(fun32,ud3,7,9);
int33 = int(fun33,ud3,9,11);
g3 = matlabFunction(int31 + int32 + int33);

pdiff3a = diff(g3,a3);
pdiff3b = diff(g3,b3);

sol3 = solve(pdiff3a,pdiff3b);

parD.a3 = double(sol3.a3);
parD.b3 = double(sol3.b3);

% Region 4
syms a4 b4 ud4
fun4 = (4.01 * (ud4-10.47).^2 + 17.79 - a4 - b4*ud4).^2;
int4 = int(fun4,ud4,11,15);
g4 = matlabFunction(int4);

pdiff4a = diff(g4,a4);
pdiff4b = diff(g4,b4);

sol4 = solve(pdiff4a,pdiff4b);

parD.a4 = double(sol4.a4);
parD.b4 = double(sol4.b4);

```

```
save('parD.mat','parD');

%% Plot real function together with approximation
% real function
step = 0.01;
ud = 0:step:15;

for i = 1:length(ud);
    if i < round(2/step)
    funreal(i) = ud(i)^2+4;
    elseif i < round(5/step)
    funreal(i) = 4*ud(i);
    elseif i < round(7/step)
    funreal(i) = -9.44*ud(i)^3+166.06*ud(i)^2-948.22*ud(i)+1790.28;
    elseif i <= round(9/step)
    funreal(i) = -11.78*ud(i) + 132.44;
    elseif i > round(9/step)
    funreal(i) = 4.01*(ud(i)-10.47)^2+17.79;
    end
end

% approximate function
for i = 1:length(ud);
    if i < round(5/step)
    funapprox(i) = parD.a1+parD.b1*ud(i);
    elseif i < round(6.5/step)
    funapprox(i) = parD.a2+parD.b2*ud(i);
    elseif i <= round(11/step)
    funapprox(i) = parD.a3+parD.b3*ud(i);
    elseif i > round(11/step)
    funapprox(i) = parD.a4+parD.b4*ud(i);
    end
end

figure;
hold on; grid on;
plot(ud,funreal);
plot(ud,funapprox);
xlabel('generated power u_d [kW]');
ylabel('consumed fuel of diesel generator [kg/h]');
legend('real fuel consumption', 'approximated fuel consumption');

% compute RMSE
rmse = rms(funreal-funapprox);
```

Exercise 2.4

```
%% SC4160 MODELLING AND CONTROL OF HYBRID SYSTEMS
% Step 2.4
% Jessie van Dam (4395832) and Miranda van Duijn (4355776)
clear all; close all; clc;
load('parD.mat');

% creating constraints
A = [0 0 0 0 0 0 0 0  0  0  1
     0 0 0 0 0 0 0 0  1 -1  0
     0 0 0 0 0 0 0 0  0  1 -1
     0 0 0 0 0 0 0 0 -1  0  0  ];

b = [15; 0; 0; 0];

% minimizing integral for several initial values
% input vector x = [a1; a2; a3; a4; b1; b2; b3; b4; u1; u2; u3]
x = zeros(20,11);
x0 = zeros(1,11);
for i = 1:21
    x0 = [parD.a1+(i-11) parD.a2+(i-11) parD.a3+(i-11) parD.a4+(i-11) parD.b1+(i-11) ...
        parD.b2+(i-11) parD.b3+(i-11) parD.b4+(i-11)...
        4+((i-11)/10) 6.5+((i-11)/10) 11+((i-11)/10)];
    x(i,:) = fmincon(@PWAapprox,x0,A,b);
    fun_int(i) = PWAapprox(x(i,:));
end

fun_min = find(fun_int == min(fun_int(:)));
x_min = x(fun_min,:);

%% Plot real function together with approximation
% real function
step = 0.01;
ud = 0:step:15;

for i = 1:length(ud);
    if i ≤ round(2/step)
    funreal(i) = ud(i)^2+4;
    elseif i ≤ round(5/step)
    funreal(i) = 4*ud(i);
    elseif i ≤ round(7/step)
    funreal(i) = −9.44*ud(i)^3+166.06*ud(i)^2−948.22*ud(i)+1790.28;
    elseif i ≤ round(9/step)
    funreal(i) = −11.78*ud(i) + 132.44;
    elseif i > round(9/step)
    funreal(i) = 4.01*(ud(i)−10.47)^2+17.79;
    end
end

% approximate function
for i = 1:length(ud);
    if i ≤ round(x_min(9)/step)
    funapprox(i) = x_min(1)+x_min(5)*ud(i);
    elseif i ≤ round(x_min(10)/step)
    funapprox(i) = x_min(2)+x_min(6)*ud(i);
    elseif i ≤ round(x_min(11)/step)
    funapprox(i) = x_min(3)+x_min(7)*ud(i);
    elseif i > round(x_min(11)/step)
    funapprox(i) = x_min(4)+x_min(8)*ud(i);
    end
end

figure;
hold on; grid on;
plot(ud,funreal);
plot(ud,funapprox);
xlabel('generated power u_d [kW]');
ylabel('consumed fuel of diesel generator [kg/h]');
legend('real fuel consumption', 'approximated fuel consumption');

% compute RMSE
rmse = rms(funreal−funapprox);
```

Exercise 2.7

```
%% SC4160 MODELLING AND CONTROL OF HYBRID SYSTEMS
% Step 2.7
% Jessie van Dam (4395832) and Miranda van Duijn (4355776)
clear all; close all; clc;
addpath C:\Users\Miranda\hysdel−2.0.6−MINGW32_NT−5.1−i686
addpath(genpath('C:\Documenten\TU Delft\MSc Systems and Control\Q4\Modelling and Control of ...
    Hybrid Systems\Project\Modelling_and_Control_of_Hybrid_Systems'))

% Define parameters
parB.eta_c = [0.9 0.95];
parB.eta_d = [0.8 0.77];
parB.x_up  = [48 64];
parB.u_low = [−3 −4];
parB.u_up  = [2 3];
parB.A     = 1;
parB.x0    = 10;

save('parB.mat','parB')

```

```
%%
load parD.mat
parD.x_low = 10;
parD.x_up  = 120;
parD.u_up  = 15;
parD.u_low = 0;
parD.Rf    = 0.4;
parD.x0    = 50;

parD.u1    = 5;
parD.u2    = 6.5;
parD.u3    = 11;

save('parD.mat','parD')

%%
dim.Ts     = 0.20;  % in [h]
dim.t      = 10;
dim.Np     = 25;    % prediction horizon
dim.Nc     = 25;    % control horizon
dim.Wb1    = 3;     % weight in cost function battery 1
dim.Wb2    = 4;     % weight in cost function battery 2
dim.Wd     = 10;    % weight in cost function diesel generator
dim.Wfuel  = 4;     % weight in cost function fuel
dim.We     = 0.4;   % weight in cost function e?

save('dim.mat','dim')

%% Defining battery with matrices
% Defining MLD matrices battery 1
MLDB1.A  = 1;
MLDB1.B1 = -dim.Ts*parB.eta_d(1);
MLDB1.B2 = 0;
MLDB1.B3 = dim.Ts*(parB.eta_d(1)-parB.eta_c(1));
MLDB1.B4 = 0;

MLDB1.E1 = [0; 0; 0; 0; -1; 1; 0; 0; 0; 0]; % E1 matrix battery 1
MLDB1.E2 = [1; -1; 1; -1; 0; 0; 0; 0; -1; 1]; % E2 matrix battery 1
MLDB1.E3 = [0; 0; parB.u_up(1); parB.u_low(1)-eps; 0; 0; -parB.u_up(1); parB.u_low(1); ...
    -parB.u_low(1); parB.u_up(1)]; % E3 matrix battery 1
MLDB1.E4 = [0; 0; 0; 0; 0; 0; 1; -1; 1; -1]; % E4 matrix battery 1
MLDB1.g5 = [parB.u_up(1); -parB.u_low(1); parB.u_up(1); -eps; 0; parB.x_up(1); 0; 0; ...
    -parB.u_low(1); parB.u_up(1)]; % g5 matrix battery 1

save('MLDB1.mat','MLDB1')

% Defining MLD matrices battery 2
MLDB2.A = 1;
MLDB2.B1 = -dim.Ts*parB.eta_c(1);
MLDB2.B2 = 0;
MLDB2.B3 = dim.Ts*(parB.eta_d(1)-parB.eta_c(1));
MLDB2.B4 = 0;

MLDB2.E1 = [0; 0; 0; 0; -1; 1; 0; 0; 0; 0]; % E1 matrix battery 2
MLDB2.E2 = [1; -1; 1; -1; 0; 0; 0; 0; -1; 1]; % E2 matrix battery 2
MLDB2.E3 = [0; 0; parB.u_up(2); parB.u_low(2)-eps; 0; 0; -parB.u_up(2); parB.u_low(2); ...
    -parB.u_low(2); parB.u_up(2)]; % E3 matrix battery 2
MLDB2.E4 = [0; 0; 0; 0; 0; 0; 1; -1; 1; -1]; % E4 matrix battery 2
MLDB2.g5 = [parB.u_up(2); -parB.u_low(2); parB.u_up(2); -eps; 0; parB.x_up(2); 0; 0; ...
    -parB.u_low(2); parB.u_up(2)]; % g5 matrix battery 2

save('MLDB2.mat','MLDB2')

%% Defining MLD system diesel generator Jordan
MLDD.A = 1;
MLDD.B1 = zeros(1,1);
MLDD.B2 = dim.Ts*[-parD.a1 -parD.a2 -parD.a3 -parD.a4];
MLDD.B3 = dim.Ts*[-parD.b1 -parD.b2 -parD.b3 -parD.b4];
MLDD.B4 = dim.Ts*parD.Rf;

% Constraint matrices diesel generator
MLDD.E1 = [0 0 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';
MLDD.E2 = [1 -1 0  0 0 0 0 -1 1 -1 1 0 0 -1 1 -1 1 0 0 -1 1 -1 1 0 0 -1 1 -1 1]';
MLDD.E3 = [0  0  1  0  0 -parD.u_up parD.u_low -parD.u_low parD.u_up eps -(parD.u1-eps)+parD.u_up ...
    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  0  0  0  0  0  0 -parD.u_up parD.u_low -parD.u_low parD.u_up parD.u1 ...
               -(parD.u2-eps)+parD.u_up 0  0  0  0  0  0  0  0  0  0  0  0;
           0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -parD.u_up parD.u_low -parD.u2 parD.u3 parD.u2 ...
               -(parD.u3-eps)+parD.u_up 0  0  0  0  0  0;
           0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -parD.u_up parD.u_low -parD.u_low ...
               parD.u_up parD.u3 -parD.u_up]';
MLDD.E4 = [0  0  0  0  0  1 -1  1 -1  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 -1  1 -1  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 -1  1 -1  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 -1  1 -1  0  0]';
MLDD.g5 = [parD.u_up; eps; 1; -parD.x_low; parD.x_up; 0; 0; -parD.u_low; parD.u_up; 0; ...
    parD.u_up; 0; 0; -parD.u_low; parD.u_up; 0;...
           parD.u_up; 0; 0; -parD.u_low; parD.u_up; 0; parD.u_up; 0; 0; -parD.u_low; ...
               parD.u_up; 0; 0];

save('MLDD.mat','MLDD')
```

Exercise 2.8 & 2.9 & 2.10

```
%% SC4160 MODELLING AND CONTROL OF HYBRID SYSTEMS
% Step 2.8 2.9 and 2.10
% Jessie van Dam (4395832) and Miranda van Duijn (4355776)
clear all; close all; clc;
addpath(genpath('C:\Documenten\TU Delft\MSc Systems and Control\Q4\Modelling and Control of ...
    Hybrid Systems\Project\Modelling_and_Control_of_Hybrid_Systems'));
addpath c:\gurobi811\win64\matlab\

%% Loading and defining parameters and data
load dim.mat; load MLDB1.mat; load MLDB2.mat; load MLDD.mat; load parB.mat; load parD.mat;
load M1.mat; load M2.mat; load M3.mat; load F1.mat; load F2.mat; load F3.mat;
load W1.mat; load W2.mat; load W3.mat; load W4.mat; load W5.mat; load S1.mat; load S2.mat;
load Ce.mat;

parB.x0 = 10;
parD.x0 = 50;
dim.Tend = 180; % Number of timesteps to optimize for

xb1 = zeros(1,dim.Tend);
xb2 = zeros(1,dim.Tend);
xd  = zeros(1,dim.Tend);
xb1(1) = parB.x0;
xb2(1) = parB.x0;
xd(1)  = parD.x0;
x   = [xb1; xb2; xd];
x(:,1) = [xb1(1); xb2(1); xd(1)];

%% Defining Pload
for k = 1:dim.Tend
    if k ≤ 20
        Pload(k) = 0;
    elseif k ≥ 21 && k ≤ 50
        Pload(k) = 30+2*k;
    elseif k ≥ 51
        Pload(k) = 45;
    end
end

%%
% Constructing matrices when it's not 12AM 12PM
[W1,W2,W3,F1,F2,F3,S1,S2] = constructMatrices(dim,parB,parD,MLDB1,MLDB2,MLDD,Pload,Ce);

% Constructing matrices when it's 12AM 12PM
MLDB1noon = MLDB1;
MLDB2noon = MLDB2;

MLDB1noon.E1 = [MLDB1.E1; -1];
MLDB1noon.E2 = [MLDB1.E2; 0];
MLDB1noon.E3 = [MLDB1.E3; 0];
```

```
MLDB1noon.E4 = [MLDB1.E4; 0];
MLDB1noon.g5 = [MLDB1.g5; -0.2*parB.x_up(1,1)];

MLDB2noon.E1 = [MLDB1.E1; -1];
MLDB2noon.E2 = [MLDB1.E2; 0];
MLDB2noon.E3 = [MLDB1.E3; 0];
MLDB2noon.E4 = [MLDB1.E4; 0];
MLDB2noon.g5 = [MLDB1.g5; -0.2*parB.x_up(1,2)];

[W1noon,W2noon,W3noon,F1noon,F2noon,F3noon,S1noon,S2noon] = ...
    constructMatrices(dim,parB,parD,MLDB1noon,MLDB2noon,MLDD,Pload,Ce);


%% Simulating closed-loop behaviour of the system

for k = 1:dim.Tend

%          if k == 60 | k == 120
%              k
%              % Minimize
%              %        W1 H + S1new Vnew
%              % Subject to
%              %             F1new Vnew ≤ F2new + F3new*x(k)
%              %        -H - W2new Vnew ≤ 0
%              %        -H + W2new Vnew ≤ 0
%
%              % names = {'H'; 'V'};
%
%              % Cost function to minimize
%              model.obj = [W1noon.W1 S1noon.S1new];
%              model.modelsense = 'min';
%              model.vtype = [repmat('C',3*dim.Np,1); ...
%                             repmat('B',dim.Np,1); repmat('C',dim.Np,1); ...
    repmat('S',dim.Np,1); repmat('B',dim.Np,1); repmat('C',dim.Np,1); ...
%                             repmat('S',dim.Np,1); repmat('B',4*dim.Np,1); ...
    repmat('C',dim.Np,1); repmat('S',4*dim.Np,1); repmat('C',dim.Np,1)];
%
%              % Constraints
%              model.A = sparse([zeros(size(F1noon.F1new,1),size(W1noon.W1,2)) F1noon.F1new; ...
%                                -eye(size(W2noon.W2new,1)) -W2noon.W2new; ...
%                                -eye(size(W2noon.W2new,1)) W2noon.W2new ]);
%              model.rhs = [F2noon.F2new+F3noon.F3new*x(:,k); zeros(size(W2noon.W2new,1),1); ...
    zeros(size(W2noon.W2new,1),1)];
%              model.sense = repmat('<',size(F2noon.F2new,1)+2*size(W2noon.W2new,1),1);
%
%              % Gurobi Solve
%              gurobi_write(model, 'mip1.lp');
%              params.outputflag = 0;
%              result = gurobi(model, params);
%              disp(result);

%          elseif ¬(k == 60*i)  % if it is not 12AM or 12PM
             % Minimize
             %        W1 H + S1new Vnew
             % Subject to
             %             F1new Vnew ≤ F2new + F3new*x(k)
             %        -H - W2new Vnew ≤ 0
             %        -H + W2new Vnew ≤ 0

             % names = {'H'; 'V'};

             % Cost function to minimize
             model.obj = [W1.W1 S1.S1new];
             model.modelsense = 'min';
             model.vtype = [repmat('C',3*dim.Np,1); ...
                            repmat('B',dim.Np,1); repmat('C',dim.Np,1); repmat('S',dim.Np,1); ...
                                repmat('B',dim.Np,1); repmat('C',dim.Np,1);...
                            repmat('S',dim.Np,1); repmat('B',4*dim.Np,1); ...
                                repmat('C',dim.Np,1); repmat('S',4*dim.Np,1); ...
                                repmat('C',dim.Np,1)];

             % Constraints
             model.A = sparse([zeros(size(F1.F1new,1),size(W1.W1,2)) F1.F1new; ...
                               -eye(size(W2.W2new,1)) -W2.W2new; ...
                               -eye(size(W2.W2new,1)) W2.W2new ]);
             model.rhs = [F2.F2new+F3.F3new*x(:,k); zeros(size(W2.W2new,1),1); ...
                 zeros(size(W2.W2new,1),1)];
             model.sense = repmat('<',size(F2.F2new,1)+2*size(W2.W2new,1),1);

             % Gurobi Solve
             gurobi_write(model, 'mip1.lp');
             params.outputflag = 0;
             result = gurobi(model, params);
             disp(result);

             % Optimized vector
             x_opti(:,k) = result.x;

             % Cost function at time step k
             J(k) = result.objval + S2.S2*x(:,k) + W3.W3*M3.M3;

             % Update equation
             xb1(:,k+1) = MLDB1.A*xb1(:,k) + MLDB1.B1*result.x(dim.Np+1) + ...
                 MLDB1.B3*result.x(2*dim.Np+1);
             xb2(:,k+1) = MLDB2.A*xb2(:,k) + MLDB2.B1*result.x(dim.Np+1) + ...
                 MLDB2.B3*result.x(2*dim.Np+1);
             xd(:,k+1)  = MLDD.A*xd(:,k) + MLDD.B2*result.x(6*dim.Np+1:6*dim.Np+4) + ...
                 MLDD.B3*result.x(11*dim.Np+1:11*dim.Np+4) + MLDD.B4;

             x(:,k+1) = [xb1(:,k+1);
                         xb2(:,k+1)
                         xd(:,k+1)];

%     end

end

%%
close all;

figure; hold on;
plot(xb1(1,:),'r')
plot(xb2(1,:),'--k')
plot(xd(1,:),'.-')
xticks([0 20 40 60 80 100 120 140 160 180])
xticklabels({'0','4','8','12','16','20','24','28','32','36'})
xlim([0 180])
xlabel('Time [h]'); ylabel('States [kWh] or [kg]')
set(gca,'FontSize',31)
lgd = legend('x_{b1}','x_{b2}','x_d')
lgd.FontSize = 31;
hold off;

figure; hold on;
plot(x_opti(dim.Np+1,:));
plot(x_opti(4*dim.Np+1,:));
plot(x_opti(10*dim.Np+1,:));
xticks([0 20 40 60 80 100 120 140 160 180])
xticklabels({'0','4','8','12','16','20','24','28','32','36'})
xlim([0 180])
xlabel('Time [h]'); ylabel('Control inputs')
set(gca,'FontSize',31)
lgd = legend('u_{b1}','u_{b2}','u_d')
lgd.FontSize = 31;
hold off;

%%
x_optimal = [x_opti(1,1); x_opti(dim.Np+1,1); x_opti(2*dim.Np+1,1)
             x_opti(3*dim.Np+1,1); x_opti(4*dim.Np+1,1); x_opti(5*dim.Np+1)
             x_opti(6*dim.Np+1:6*dim.Np+4,1); x_opti(10*dim.Np+1,1); ...
                 x_opti(11*dim.Np+1:11*dim.Np+4,1)];
```

Function for optimization matrices

```
function [ W1,W2,W3,F1,F2,F3,S1,S2 ] = constructMatrices( ...
    dim,parB,parD,MLDB1,MLDB2,MLDD,Pload,Ce)
%constructMatrices Summary of this function goes here
%   Detailed explanation goes here

    % Constructing M matrices for update equation MILP
    % M1.M1 matrix for the diesel generator
    M1.M1_d_∆ = zeros(dim.Np*size(MLDD.A,1),dim.Np*size(MLDD.B2,2));
    M1.M1_d_u = zeros(dim.Np,dim.Np);
    M1.M1_d_zd = zeros(dim.Np*size(MLDD.A,1),dim.Np*size(MLDD.B3,2));

    for np1 = 1:dim.Np % over columns
        for np2 = 1:dim.Np % over rows
        M1.M1_d_∆(1+(np2-1)*size(MLDD.A,1),1+(np1-1)*size(MLDD.B2,2):np1*size(MLDD.B2,2))  = ...
            MLDD.A^(np2-1)*MLDD.B2;
        M1.M1_d_u(1+(np2-1)*size(MLDD.A,1),1+(np1-1):np1*size(MLDD.B1,2)) ...
                                  = MLDD.A^(np2-1)*MLDD.B1;
        M1.M1_d_zd (1+(np2-1)*size(MLDD.A,1),1+(np1-1)*size(MLDD.B3,2):np1*size(MLDD.B3,2)) ...
                = MLDD.A^(np2-1)*MLDD.B3;
        end
    end
    clear np1

    for i = 1:dim.Np
        for j = 1:4*dim.Np
            if j > 4*i
            M1.M1_d_∆(i,j) = 0;
            M1.M1_d_zd(i,j) = 0;
            end
        end
    end
    clear i j

    M1.M1_d = [M1.M1_d_∆ M1.M1_d_u M1.M1_d_zd];

    % M1.M1 matrix for the batteries
    M1.M1_b1_∆ = zeros(dim.Np,dim.Np);
    M1.M1_b1_u     = zeros(dim.Np,dim.Np);
    M1.M1_b1_zd    = zeros(dim.Np,dim.Np);
    M1.M1_b1       = zeros(dim.Np,3*dim.Np);

    M1.M1_b2_∆ = zeros(dim.Np,dim.Np);
    M1.M1_b2_u     = zeros(dim.Np,dim.Np);
    M1.M1_b2_zd    = zeros(dim.Np,dim.Np);
    M1.M1_b2       = zeros(dim.Np,3*dim.Np);

    for np1 = 1:dim.Np % over columns
        for np2 = 1:dim.Np % over rows
        M1.M1_b1_∆(1+(np2-1)*size(MLDB1.A,1),1+(np1-1)*size(MLDB1.B2,2):np1*size(MLDB1.B2,2)) ...
             = MLDB1.A^(np2-1)*MLDB1.B2;
        M1.M1_b1_u(1+(np2-1)*size(MLDB1.A,1),1+(np1-1)*size(MLDB1.B3,2):np1*size(MLDB1.B3,2)) ...
                 = MLDB1.A^(np2-1)*MLDB1.B1;
        M1.M1_b1_zd(1+(np2-1)*size(MLDB1.A,1),1+(np1-1)*size(MLDB1.B1,2):np1*size(MLDB1.B1,2)) ...
                = MLDB1.A^(np2-1)*MLDB1.B3;

        M1.M1_b2_∆(1+(np2-1)*size(MLDB2.A,1),1+(np1-1)*size(MLDB2.B2,2):np1*size(MLDB2.B2,2)) ...
             = MLDB2.A^(np2-1)*MLDB2.B2;
        M1.M1_b2_u(1+(np2-1)*size(MLDB2.A,1),1+(np1-1)*size(MLDB2.B3,2):np1*size(MLDB2.B3,2)) ...
                 = MLDB2.A^(np2-1)*MLDB2.B1;
        M1.M1_b2_zd(1+(np2-1)*size(MLDB2.A,1),1+(np1-1)*size(MLDB2.B1,2):np1*size(MLDB2.B1,2)) ...
                = MLDB2.A^(np2-1)*MLDB2.B3;
        end
    end
    clear np1 np2

    for i = 1:dim.Np
        for j = 1:dim.Np
            if j > i
            M1.M1_b1_∆(i,j) = 0;
            M1.M1_b1_u(i,j)     = 0;
            M1.M1_b1_zd(i,j)    = 0;

            M1.M1_b2_∆(i,j) = 0;
            M1.M1_b2_u(i,j)      = 0;
            M1.M1_b2_zd(i,j)     = 0;
            end
        end
    end
    clear i j

    M1.M1_b1 = [M1.M1_b1_Δ M1.M1_b1_u M1.M1_b1_zd];
    M1.M1_b2 = [M1.M1_b2_Δ M1.M1_b2_u M1.M1_b2_zd];

    % Total M1.M1 matrix
    M1.M1 = [M1.M1_b1 zeros(dim.Np,size(M1.M1_b2,2)) zeros(dim.Np,size(M1.M1_d,2)); ...
        zeros(dim.Np,size(M1.M1_b1,2)) M1.M1_b2 zeros(dim.Np,size(M1.M1_d,2)); ...
        zeros(dim.Np,size(M1.M1_b1,2)) zeros(dim.Np,size(M1.M1_b2,2)) M1.M1_d];

    % M2.M2 matrix
    M2.M2_b1 = zeros(dim.Np,size(MLDB1.A,2));
    M2.M2_b2 = zeros(dim.Np,size(MLDB1.A,2));
    M2.M2_d  = zeros(dim.Np,size(MLDD.A,2));

    for np = 1:dim.Np
        M2.M2_b1(np,:)  =  (MLDB1.A)^np;
        M2.M2_b2(np,:)  =  (MLDB2.A)^np;
        M2.M2_d(np,:)   =  (MLDD.A)^np;
    end
    clear np

    M2.M2 = [M2.M2_b1 zeros(size(M2.M2_b1,1),size(M2.M2_b2,2)) ...
        zeros(size(M2.M2_b1,1),size(M2.M2_d,2)); ...
             zeros(size(M2.M2_b2,1),size(M2.M2_b1,2)) M2.M2_b2 ...
                 zeros(size(M2.M2_b2,1),size(M2.M2_d,2)); ...
             zeros(size(M2.M2_d,1),size(M2.M2_b1,2)) zeros(size(M2.M2_d,1),size(M2.M2_b2,2)) ...
                 M2.M2_d];

    % M3.M3 matrix
    M3.M3_b1 = zeros(dim.Np*size(MLDD.B4,1),size(MLDD.B4,2));
    M3.M3_b2 = zeros(dim.Np*size(MLDD.B4,1),size(MLDD.B4,2));
    M3.M3_d  = zeros(dim.Np*size(MLDD.B4,1),size(MLDD.B4,2));

    for nd = 1:dim.Np
        if nd == 1
            M3.M3_d(1,1)  = MLDD.B4;
        end

        if nd > 1
            M3.M3_d(1+(nd-1)*size(MLDD.B4,1):nd*size(MLDD.B4,1),1:size(MLDD.B4,2)) = ...
                M3.M3_d(nd-1,:) + MLDD.B4*(MLDD.A)^(nd-1);
        end
    end
    clear nd

    % Concatenating submatrices into complete M3.M3 matrix
    M3.M3 = [M3.M3_b1; M3.M3_b2; M3.M3_d];

    % Constructing F1.F1 matrix for MILP constraint equation
    F1.F1b1_Δ = zeros(dim.Np*size(MLDB1.E3,1),dim.Np*size(MLDB1.E3,2));
    F1.F1b1_u = zeros(dim.Np*size(MLDB1.E2,1),dim.Np*size(MLDB1.E2,2));
    F1.F1b1_zd = zeros(dim.Np*size(MLDB1.E4,1),dim.Np*size(MLDB1.E4,2));

    F1.F1b2_Δ = zeros(dim.Np*size(MLDB2.E3,1),dim.Np*size(MLDB2.E3,2));
    F1.F1b2_u = zeros(dim.Np*size(MLDB2.E2,1),dim.Np*size(MLDB2.E2,2));
    F1.F1b2_zd = zeros(dim.Np*size(MLDB2.E4,1),dim.Np*size(MLDB2.E4,2));

    F1.F1d_Δ = zeros(dim.Np*size(MLDD.E3,1),dim.Np*size(MLDD.E3,2));
    F1.F1d_u = zeros(dim.Np*size(MLDD.E2,1),dim.Np*size(MLDD.E2,2));
    F1.F1d_zd = zeros(dim.Np*size(MLDD.E3,1),dim.Np*size(MLDD.E4,2));

    for np1 = 1:dim.Np % over columns
        for np2 = 1:dim.Np % over rows
            if np1 == np2
            % For battery 1
            F1.F1b1_Δ(1+(np2-1)*size(MLDB1.E3,1):np2*size(MLDB1.E3,1),...
                                  1+(np1-1)*size(MLDB1.E3,2):np1*size(MLDB1.E3,2))     = MLDB1.E3;
            F1.F1b1_u(1+(np2-1)*size(MLDB1.E2,1):np2*size(MLDB1.E2,1),...
                          1+(np1-1)*size(MLDB1.E2,2):np1*size(MLDB1.E2,2))       = MLDB1.E2;
            F1.F1b1_zd(1+(np2-1)*size(MLDB1.E4,1):np2*size(MLDB1.E4,1),...
                          1+(np1-1)*size(MLDB1.E4,2):np1*size(MLDB1.E4,2))       = MLDB1.E4;

            % For battery 2
            F1.F1b2_Δ(1+(np2-1)*size(MLDB2.E3,1):np2*size(MLDB2.E3,1),...
                              1+(np1-1)*size(MLDB2.E3,2):np1*size(MLDB2.E3,2))   = MLDB2.E3;
            F1.F1b2_u(1+(np2-1)*size(MLDB2.E2,1):np2*size(MLDB2.E2,1),...
                          1+(np1-1)*size(MLDB2.E2,2):np1*size(MLDB2.E2,2))       = MLDB2.E2;
            F1.F1b2_zd(1+(np2-1)*size(MLDB2.E4,1):np2*size(MLDB2.E4,1),...
                           1+(np1-1)*size(MLDB2.E4,2):np1*size(MLDB2.E4,2))      = MLDB2.E4;

            % For the diesel generator
            F1.F1d_Δ(1+(np2-1)*size(MLDD.E3,1):np2*size(MLDD.E3,1),...
                             1+(np1-1)*size(MLDD.E3,2):np1*size(MLDD.E3,2))      = MLDD.E3;
            F1.F1d_u(1+(np2-1)*size(MLDD.E2,1):np2*size(MLDD.E2,1),...
                         1+(np1-1)*size(MLDD.E2,2):np1*size(MLDD.E2,2))          = MLDD.E2;
            F1.F1d_zd(1+(np2-1)*size(MLDD.E4,1):np2*size(MLDD.E4,1),...
                          1+(np1-1)*size(MLDD.E4,2):np1*size(MLDD.E4,2))         = MLDD.E4;
            end

            if np2 > np1 % to fill only below the block diagonal
            % NOTE: algemener maken voor F1.F1_bi_Δ?
            % For battery 1
            F1.F1b1_u(1+(np2-1)*size(MLDB1.E2,1):np2*size(MLDB1.E2,1),...
                          1+(np1-1)*size(MLDB1.E2,2):np1*size(MLDB1.E2,2))       = ...
                              MLDB1.E1*MLDB1.A^(np2-2)*MLDB1.B1;
            F1.F1b1_zd(1+(np2-1)*size(MLDB1.E4,1):np2*size(MLDB1.E4,1),...
                           1+(np1-1)*size(MLDB1.E4,2):np1*size(MLDB1.E4,2))      = ...
                               MLDB1.E1*MLDB1.A^(np2-2)*MLDB1.B3;

            % For battery 2
            F1.F1b2_u(1+(np2-1)*size(MLDB2.E2,1):np2*size(MLDB2.E2,1),...
                          1+(np1-1)*size(MLDB2.E2,2):np1*size(MLDB2.E2,2))       = ...
                              MLDB2.E1*MLDB2.A^(np2-2)*MLDB2.B1;
            F1.F1b2_zd(1+(np2-1)*size(MLDB2.E4,1):np2*size(MLDB2.E4,1),...
                           1+(np1-1)*size(MLDB2.E4,2):np1*size(MLDB2.E4,2))      = ...
                               MLDB2.E1*MLDB2.A^(np2-2)*MLDB2.B3;

            % For the diesel generator
            F1.F1d_Δ(1+(np2-1)*size(MLDD.E3,1):np2*size(MLDD.E3,1),...
                             1+(np1-1)*size(MLDD.E3,2):np1*size(MLDD.E3,2))      = ...
                                 MLDD.E1*MLDD.A^(np2-2)*MLDD.B2;
            F1.F1d_zd(1+(np2-1)*size(MLDD.E4,1):np2*size(MLDD.E4,1),...
                          1+(np1-1)*size(MLDD.E4,2):np1*size(MLDD.E4,2))         = ...
                              MLDD.E1*MLDD.A^(np2-2)*MLDD.B3;
            end
        end
    end
    clear np1 np2

    % Concatenating all submatrices
    F1.F1b1 = [F1.F1b1_Δ F1.F1b1_u F1.F1b1_zd];
    F1.F1b2 = [F1.F1b2_Δ F1.F1b2_u F1.F1b2_zd];
    F1.F1d = [F1.F1d_Δ F1.F1d_u F1.F1d_zd];
    F1.F1 = [F1.F1b1 zeros(size(F1.F1b1,1),size(F1.F1b2,2)) ...
        zeros(size(F1.F1b1,1),size(F1.F1d,2));
             zeros(size(F1.F1b2,1),size(F1.F1b1,2)) F1.F1b2 zeros(size(F1.F1b2,1),size(F1.F1d,2))
             zeros(size(F1.F1d,1),size(F1.F1b1,2)) zeros(size(F1.F1d,1),size(F1.F1b2,2)) F1.F1d];

    % Constructing F1 matrices for Pimp
    F1.F11b1 = zeros(dim.Np,3*dim.Np);
    F1.F11d  = zeros(dim.Np,9*dim.Np);

    % F1 for the batteries
    diag = [0 1 0];
    F1.F11b1 = kron(eye(dim.Np),diag);
    F1.F11b2 = F1.F11b1;

    % F1 for the diesel generator
    diagd = [0  0  0  0  1  0  0  0  0];
    F1.F11d = kron(eye(dim.Np),diagd);

    F1.F1new = [F1.F1 zeros(size(F1.F1,1),dim.Np);
                F1.F11b1 F1.F11b2 F1.F11d eye(dim.Np);
                -F1.F11b1 -F1.F11b2 -F1.F11d -eye(dim.Np)];

    % Constructing F2.F2 matrix for MILP constraint equation
    F2.F2b1   = zeros(dim.Np*size(MLDB1.g5,1),size(MLDB1.g5,2));
    F2.F2b2   = zeros(dim.Np*size(MLDB2.g5,1),size(MLDB2.g5,2));
    F2.F2d_1  = zeros(dim.Np*size(MLDD.g5,1),size(MLDD.g5,2));
    F2.F2d_2  = zeros(dim.Np*size(MLDD.g5,1),size(MLDD.g5,2));

    for n = 1:dim.Np
        F2.F2b1(1+(n-1)*size(MLDB1.g5,1):n*size(MLDB1.g5,1),1:size(MLDB1.g5,2)) = MLDB1.g5;
        F2.F2b2(1+(n-1)*size(MLDB2.g5,1):n*size(MLDB2.g5,1),1:size(MLDB2.g5,2)) = MLDB2.g5;
        F2.F2d_1(1+(n-1)*size(MLDD.g5,1):n*size(MLDD.g5,1),1:size(MLDD.g5,2)) = MLDD.g5;

        if n == 2
            F2.F2d_2(1+(n-1)*size(MLDD.E1):n*size(MLDD.E1,1),1:size(MLDD.g5,2)) = ...
                MLDD.E1*MLDD.B4;
        end

        if n > 2
            F2.F2d_2(1+(n-1)*size(MLDD.E1):n*size(MLDD.E1,1),1:size(MLDD.g5,2)) = ...
                F2.F2d_2(1+(n-2)*size(MLDD.E1,1):(n-1)*size(MLDD.E1,1)) + ...
                    MLDD.E1*MLDD.B4*(MLDD.A)^(n-2);
        end

    end
    clear n

    % Concatenating submatrices into complete F2.F2 matrix
    F2.F2d = F2.F2d_1 - F2.F2d_2;
    F2.F2 = [F2.F2b1; F2.F2b2; F2.F2d];

    F2.F2new = [F2.F2;
                Pload(1:dim.Np)';
                -Pload(1:dim.Np)'];

    % Constructing F3.F3 matrix for MILP constraint equation
    F3.F3b1 = zeros(dim.Np*size(MLDB1.E1,1),size(MLDB1.E1,2));
    F3.F3b2 = zeros(dim.Np*size(MLDB2.E1,1),size(MLDB2.E1,2));
    F3.F3d  = zeros(dim.Np*size(MLDD.E1,1),size(MLDD.E1,2));

    for n = 1:dim.Np
        F3.F3b1(1+(n-1)*size(MLDB1.E1,1):n*size(MLDB1.E1,1),1:size(MLDB1.E1,2)) = ...
            -MLDB1.E1*MLDB1.A^(n-1);
        F3.F3b2(1+(n-1)*size(MLDB2.E1,1):n*size(MLDB2.E1,1),1:size(MLDB2.E1,2)) = ...
            -MLDB2.E1*MLDB2.A^(n-1);
        F3.F3d(1+(n-1)*size(MLDD.E1,1):n*size(MLDD.E1,1),1:size(MLDD.E1,2))     = ...
            -MLDD.E1*MLDD.A^(n-1);
    end
    clear n

    F3.F3 = [F3.F3b1 zeros(size(F3.F3b1,1),size(F3.F3b2,2)) ...
        zeros(size(F3.F3b1,1),size(F3.F3d,2));
          zeros(size(F3.F3b2,1),size(F3.F3b1,2)) F3.F3b2 zeros(size(F3.F3b2,1),size(F3.F3d,2))
          zeros(size(F3.F3d,1),size(F3.F3b1,2)) zeros(size(F3.F3d,1),size(F3.F3b2,2)) F3.F3d];

    F3.F3new = [F3.F3;
                zeros(2*dim.Np,3)];

    % Constructing W matrices for optimization
    W1.W1b1 = [dim.Wb1*ones(1,dim.Np-1) 0];
    W1.W1b2 = [dim.Wb2*ones(1,dim.Np-1) 0];
    W1.W1d = [dim.Wd*ones(1,dim.Np-1) 0];
    W1.W1 = [ W1.W1b1 W1.W1b2 W1.W1d ];

    % W2 matrices again using submatrices
    W2.W2b1 = zeros(dim.Np,3*dim.Np);
    W2.W2b2 = zeros(dim.Np,3*dim.Np);
    W2.W2d  = zeros(dim.Np,9*dim.Np);

    for nr = 1:dim.Np
```

```
        for nc = 1:3*dim.Np
            if nr == nc
                if nc ≤ dim.Np
                    W2.W2b1(nr,nc) = 1;
                    W2.W2b2(nr,nc) = 1;
                end
            end

            if  nr == nc+1
                    W2.W2b1(nr,nc) = −1;
                    W2.W2b2(nr,nc) = −1;
            end
        end

        for nc = 1:9*dim.Np
            if nc == 1+(nr−1)*4
                W2.W2d(nr,nc:nc+3) = 1;
            end

        end

        for nc = 1:9*dim.Np−1
            if nc == 1+(nr−1)*4
                W2.W2d(nr+1,nc:nc+3) = −1;
            end
        end

    end

    W2.W2d = W2.W2d(1:end−1,:);

    W2.W2 = [W2.W2b1 zeros(size(W2.W2b1,1),size(W2.W2b2,2)) ...
        zeros(size(W2.W2b1,1),size(W2.W2d,2)); ...
             zeros(size(W2.W2b2,1),size(W2.W2b1,2)) W2.W2b2 ...
                 zeros(size(W2.W2b2,1),size(W2.W2d,2)); ...
             zeros(size(W2.W2d,1),size(W2.W2b1,2)) zeros(size(W2.W2d,1),size(W2.W2b2,2)) W2.W2d];

    W2.W2new = [W2.W2 zeros(size(W2.W2,1),dim.Np)];

    % W3 matrices
    W3.W3b1 = [zeros(1,dim.Np−1) −dim.We];
    W3.W3b2 = [zeros(1,dim.Np−1) −dim.We];
    W3.W3d  = [zeros(1,dim.Np−1) −dim.Wfuel];
    W3.W3 = [W3.W3b1 W3.W3b2 W3.W3d];

    % W4 matrices
    W4.W4b1 = dim.We;
    W4.W4b2 = dim.We;
    W4.W4d  = dim.Wfuel;
    W4.W4 = [W4.W4b1 W4.W4b2 W4.W4d];

    % W5 matrices
    W5.W5b1 = [zeros(1,dim.Np) −Ce(1,1:dim.Np−1) 0 zeros(1,dim.Np)];
    W5.W5b2 = [zeros(1,dim.Np) −Ce(1,1:dim.Np−1) 0 zeros(1,dim.Np)];
    W5.W5d  = [zeros(1,4*dim.Np) −Ce(1,1:dim.Np−1) 0 zeros(1,4*dim.Np)];

    W5.W5 = [W5.W5b1 W5.W5b2 W5.W5d];

    % S matrices
    S1.S1b1 = W3.W3b1*M1.M1_b1+W5.W5b1;
    S1.S1b2 = W3.W3b2*M1.M1_b2+W5.W5b2;
    S1.S1d  = W3.W3d*M1.M1_d+W5.W5d;

    S1.S1 = W3.W3*M1.M1+W5.W5;

    S1.S1b1new = [W3.W3b1*M1.M1_b1 Ce(1:dim.Np−1) 0];
    S1.S1b2new = [W3.W3b2*M1.M1_b2 Ce(1:dim.Np−1) 0];
    S1.S1dnew  = [W3.W3d*M1.M1_d Ce(1:dim.Np−1) 0];

    S1.S1new = [W3.W3*M1.M1+W5.W5 Ce(1:dim.Np−1) 0];

    S2.S2b1 = W3.W3b1*M2.M2_b1*W4.W4b1;
    S2.S2b2 = W3.W3b2*M2.M2_b2*W4.W4b2;
    S2.S2d  = W3.W3d*M2.M2_d*W4.W4d;
    S2.S2   = W3.W3*M2.M2+W4.W4;

end
```