UNIVERSITÄT KAISERSLAUTERN

Seminararbeit

Energy Management Strategies for parallel Hybrid Electric Vehicle

Krunalkumar Zadafiya

Matrikel Nr.: 415657

Subject Module: Seminar Electromobility

Supervisor: APL. PROF. DR.-ING. DANIEL GÖRGES

0. Summary

Hybrid vehicles are becoming more and more popular over time, and the development of new techniques to optimise them is now a common topic in the engineering world. A hybrid vehicle has many advantages, such as lower fuel consumption and CO2 emissions, but should be optimised with suitable software. MATLAB/Simulink is now a readily available software for optimising the hybrid drive system. In addition, there are also some challenges such as energy management, phlegmatization etc.

The aim of this project is to implement the best strategy for minimizing the fuel consumption and maintaining the battery charge of Mild-Hybrid electric vehicle. The initial parameters of a mild-Hybrid electric vehicle are given. The model-based system for calculation of fuel consumption namely Qss_Toolbox is also available. The task of this seminar is to use any optimization strategy and implement it for the given vehicle. To find the best possible result, I have used the Dynamic Programming and Rule Based strategy. The optimization with the Rule Based strategy is easy, but not optimum. Whereas the dynamic programming strategy needs in depth understanding of model and different driving modes.

Once the complete model and driving modes are studied, the implementation Dynamic programming and Rule based strategy is started.

1. Optimization with Rule-Based strategy

Using MATLAB environment, I have got different torque range in which different driving modes can be adopted. Qss_Toolbox in Simulink environment is shown in the figure:

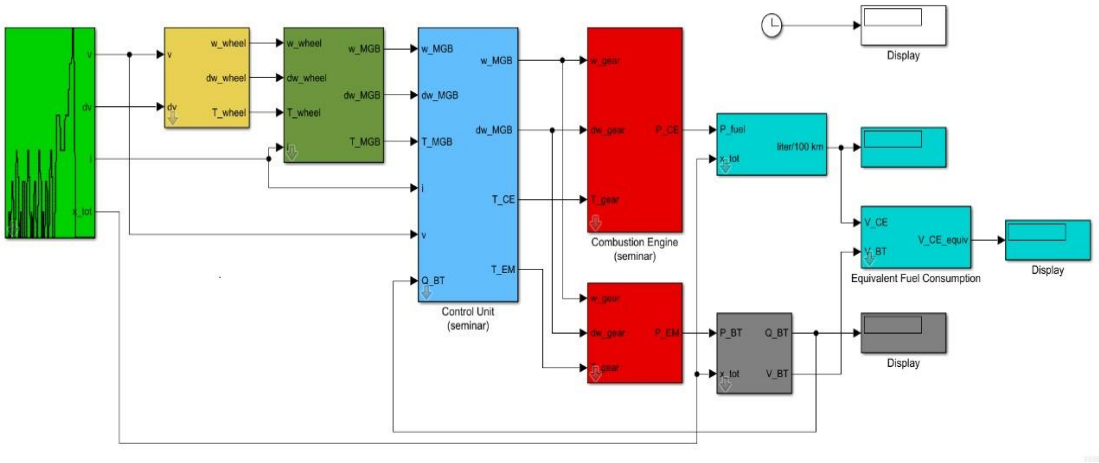

Fig. 1. Block diagram from Qss_Toolbox

After implementing the vehicle parameters, the strategy implementation starts.

The code for optimization for Rule Based strategy is given below:

```
%% Setup of given Model
filepath = 'qss_hybrid_electric_vehicle.slx';

% Load simulink model
load_system(filepath);

% set NEDC cycle
set_param(strcat(filepath(1:end-4),'/Driving Cycle'),'cyclenr', 7);

% initialize the array in which consumption will be stored
Consumption_Curve_NEDC = [];
Battery_Charge_NEDC = [];


%% Paramater setup for NEDC cycle
T_MGB_th_lower_NEDC = 10;
T_MGB_th_upper_NEDC = 100;
T_MGB_TH_NEDC = T_MGB_th_lower_NEDC:1:T_MGB_th_upper_NEDC;

T_lower_NEDC = 10;
T_upper_NEDC = 40;
T_TH_NEDC = T_lower_NEDC:1:T_upper_NEDC;

u_LPS_min_lower_NEDC = -0.1;
u_LPS_min_upper_NEDC = -0.9;
U_LPS_MIN_NEDC = u_LPS_min_lower_NEDC:-0.01:u_LPS_min_upper_NEDC;
```

```
[X_NEDC, Y_NEDC, Z_NEDC] = meshgrid(T_MGB_TH_NEDC, T_TH_NEDC,...
                                    ...U_LPS_MIN_NEDC);


%% Run Simulation : NEDC
for x = 1:1:size(X_NEDC,1)
    for y = 1:1:size(Y_NEDC,2)
        for z = 1:1:size(Z_NEDC,3)
            T_MGB_th = X_NEDC(x,y,z);
            T = Y_NEDC(x,y,z);
            u_LPS_min = Z_NEDC(x,y,z);
            outputs = sim(filepath,'StartTime','0',...
                          ...'StopTime','1220', 'FixedStep', '1');

            if (round(outputs.q_BT(end),3) >= 0.499 &&...
                    ...round(outputs.q_BT(end),3) <= 0.500 &&...
                    ... length(outputs.q_BT)==1220)

                Battery_Charge_NEDC(x,y,z) = outputs.q_BT(end);
                Consumption_Curve_NEDC(x,y,z) =...
                          ...outputs.V_CE_equiv(end);
            else
                Battery_Charge_NEDC(x,y,z) = Inf;
                Consumption_Curve_NEDC(x,y,z) = Inf;
            end
        end
    end
end

[val, loc] = min(Consumption_Curve_NEDC(Consumption_Curve_NEDC> 0));
[xx, yy, zz] = ind2sub(size(Consumption_Curve_NEDC), loc);

fprintf('-----------------------------------------------------\n');
fprintf('T_MGB_th = %s Nm\n',num2str(X_NEDC(xx,yy,zz)));
fprintf('T = %s Nm\n',num2str(Y_NEDC(xx,yy,zz)));
fprintf('u_LPS_min = %s\n', num2str(Z_NEDC(xx,yy,zz)));
fprintf('-----------------------------------------------------\n');
```

At the end of this simulation, the value of T_MGB_th, T and u_LPS_min give optimum fuel consumption and at the same time, it sustains the battery charge.

The resulting data can be found in the table 1.

TABLE 1.
RESULTS OF RULE BASED STRATEGY

RBS	Equivalent Fuel consumption in l/100km	Charge sustainment	Percentage saving compared to convention vehicle
NEDC	3.56	18010 (yes)	27.3%
FTP-75	3.283	18010 (yes)	29.77%

The fuel consumption of Hybrid electric vehicle can be optimized using the DPM function [1]. This DPM function calculates the cost function which is the fuel consumption for this seminar and minimize it by satisfying all the boundary condition and by avoiding all the infeasibilities.

There are plenty of parameters which can be initialized to run the respected simulation and to get the respective results. Thus, considering the given task, I have chosen the torque split ratio and battery charge for the grid parameters. This will be stored in the hev.m file.

The hev_main.m function file includes modelling equations. Here, some challenging tasks take place. I have used Simulink file for the calculation. The hev_main.m function file include loading and calling function of Simulink file. The same calculation can also be done using MATLAB and it will be stored in the same file instead of calling the Simulink file.

The Simulink model is shown in Fig. 1.

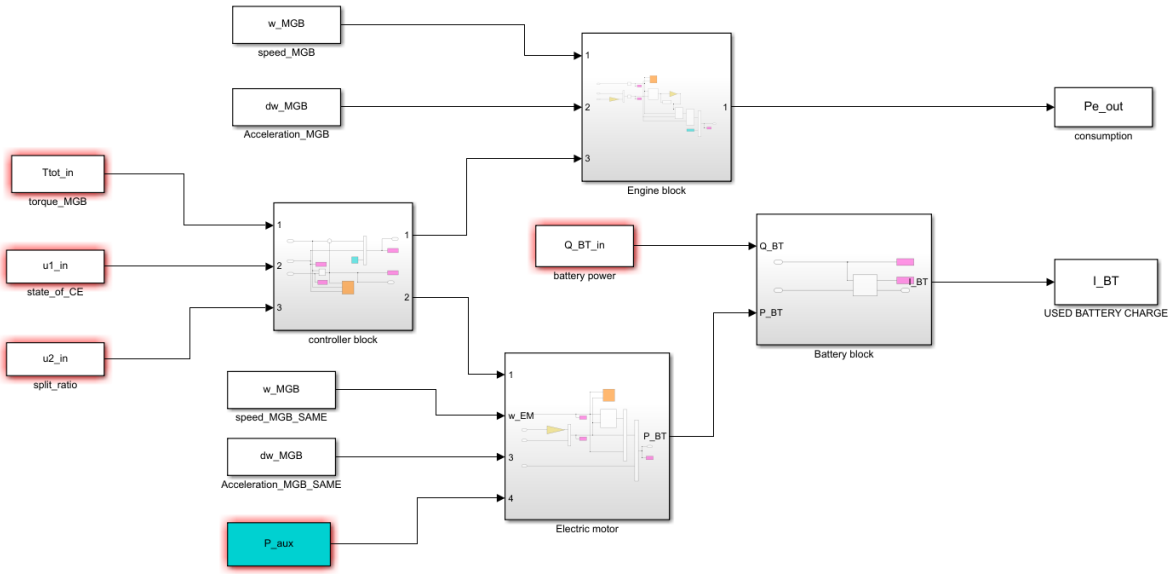

Fig. 1. Simulink model for calculation of fuel consumption for DPM function

At the end of this simulation, the torque split ratio can be found out. This type of distribution gives optimum fuel consumption as well as sustain the battery charge.

The resulting data from the Dynamic Programming can be found in the table 2.

TABLE 2.
RESULTS OF DYNAMIC PROGRAMMING

DPM	Equivalent Fuel consumption in l/100km	Charge sustainment	Percentage saving compared to convention vehicle
NEDC	3.27	18010 (yes)	33.22%
FTP-75	No feasible solution	--	--

Here, DPM function gives infeasible solution for FTP-75 cycle because there are lot more chances to get the battery discharge or to not sustain the battery charge at the end. NEDC cycle is shorter and smother than the FTP-75 cycle, so it gives benchmarking result. The percentage of fuel saving compared to the conventional vehicle is 33.22%, which is much better than the RBS. Further, Dynamic Programming strategy can be improvised by using more efficient and more discrete parameters.

Rule based strategy is ideal to imply directly in the vehicle, but not so much efficient as Dynamic programming. Taking the initial idea from performance of dynamic programming, we can make the rule-based strategy better. Dynamic programming is very efficient; however, it is hard to implement in the vehicle. Moreover, our DPM function is applicable for the FTP-75 cycle. So, the future work could be to implement this strategy with higher battery capacity. Further, using the more initial parameters and different way to initialize the parameter could be the option to solve the problem coming in to DPM.

References

[1] O. Sundstrom and L. Guzzella, "A generic dynamic programming Matlab function," 2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC), St. Petersburg, 2009, pp. 1625-1630, doi: 10.1109/CCA.2009.5281131.

Seminararbeit

PROBLEM AND SOLUTIONS

Energy Management Strategies for parallel Hybrid Electric Vehicle

Krunalkumar Zadafiya

Matrikel Nr.: 415657

Rule-Based strategy:

Problem 1: After making the grid, we find the consumption for all the values. The task was to get the minimum fuel consumption, where the battery charge is sustained. The problem was found out that the minimum fuel consumption is become zero.

Solution: Battery charge sustainment can be analysed by checking the last value of q_BT. Minimum fuel consumption will obviously not be zero. This could happen because in the range where driving modes are defined, our system gives infeasible solution. This problem can be solved by checking whether the complete cycle is run or not.

```
if (round(outputs.q_BT(end),3) >= 0.499 &&...
              ...round(outputs.q_BT(end),3) <= 0.500 &&...
              ... length(outputs.q_BT)==1220)

    Battery_Charge_NEDC(x,y,z) = outputs.q_BT(end);
    Consumption_Curve_NEDC(x,y,z) =...
                   ...outputs.V_CE_equiv(end);
else
    Battery_Charge_NEDC(x,y,z) = Inf;
    Consumption_Curve_NEDC(x,y,z) = Inf;
end
```

Dynamic programming:

Problem 2: The main task is to implement the qss_toolbox in the dynamic programming function and to initialize parameters. Depending upon the initialized parameters and calculations in the Simulink file, drive mode is being changed. These initialized parameters are possibly able to drive the vehicle in electric, engine, regenerative braking and in LPS mode. At some point, dynamic programming cannot satisfy the initial requirement to run these different modes.

Solution: To satisfy all the initial requirement to run different mode, following equations can be taken into considerations:

```
% Dynamic parameters
u1 =(Ttot>0).*(inp.U{1}<1);     %engine start-stop
u2 = inp.U{1};                  %split ratio
```

Further to minimize the calculation time, following code can be replaced:

```
if (Ttot>=0)
    Tm = Ttot.*(inp.U{1});          % torque of motor
    Te = Ttot.*(1 - inp.U{1});      % torque of engine
else
    Tm = ones(size(inp.U{1})).*Ttot.*...
            ...(min((interp1(w_EM_max,-T_EM_max,wg)...
            ...+abs(theta_EM*dwg)+epsilon)/Ttot,1));
    Te = ones(size(inp.U{1})).*(-Inf);
end
```

Last but not least, substitution for this problem can be:

In hev.m file:

```
%engine start-stop function
grd.Nu{2} = 2;
grd.Un{2}.hi = 1;
grd.Un{2}.lo = 0;
```

In hev_main.m file:

```
%engine start-stop function
u1 = inp.U{2};                %engine start-stop
u2 = inp.U{1};                %split ratio
```

Problem 3: The dynamic programming using Simulink takes too much time to simulate. What could be the solution?

Solution: The first necessary action is to understand the Simulink and MATLAB well. My first successfully simulated DP function has taken around 30 hours, which was reduced to 3 hours by making some changes in the Simulink and MATLAB file. However, it is much longer. The real-life solution is to convert the Simulink modelling into the MATLAB coding. After this seminar work, I have implemented my new DP function and it now takes 15 minutes. Further, possible try is to reduce it by avoiding unnecessary calculations.

There are lots of stakes in this simulation. Problem solving for the same can be done using the conditional brake point by right clicking on the left-hand side column of respected line of code. To analyse the system to become infeasible, following code is useful, which is already published by [1].

```
if numel(find(I==0))==0
    %condition check for infeasible system.
    keyboard
end
```

References

[1] O. Sundstrom and L. Guzzella, "A generic dynamic programming Matlab function," 2009 IEEE Control Applications, (CCA) & Intelligent Control, (ISIC), St. Petersburg, 2009, pp. 1625-1630, doi: 10.1109/CCA.2009.5281131.